Lecture Week 10

ECEN 5613 Embedded System Design
Linden McClure, Ph.D.
10/30/2023

Electrical, Computer, and Energy Engineering Department
University of Colorado Boulder

Outline

Topics

- Syllabus Review
- Debugging
- Grading
- Lab #4 Topics – I2C, LCD

Elements of ECEN 5613 still pending

- Lectures
- Office Hours
- Lab #4 work and sign-offs
- Final Project work, demo, report
- Student return of all signed out items – hard deadline is Saturday 12/16

Week 9: October 23 *(PDR presentations, Lab #3 Part 3 elements and submission due this week)*

- Final Project Design Review (PDR). Each project team presents development plan and milestones.
- Lab #4 topics. I^2C, EEPROMs, synchronous serial communication, debugging techniques.

Week 10: October 30 *(Finish Lab #4 Part 1 elements this week)*

- Final Project Design Review (PDR) – any remaining presentations.
- Lab #4 topics – LCDs, SPI. Work on Lab #4 and final project.

Week 11: November 6 *(Finish Lab #4 Part 2 elements this week)*

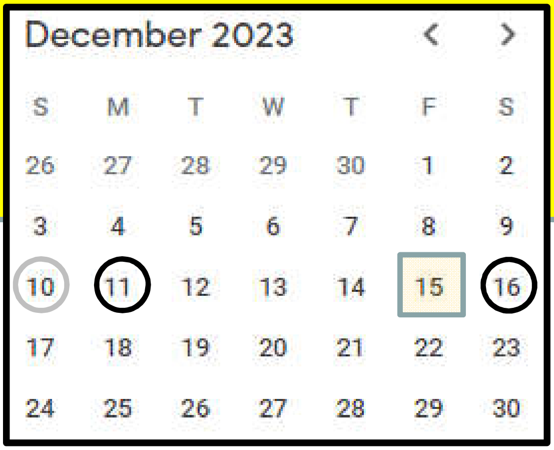
- Data Conversion - Analog-to-Digital Converters (ADCs), Digital-to-Analog Converters (DACs).
- Passive components. Firmware design, main loop/interrupt driven designs, device drivers.

Week 12: November 13 *(Lab #4 Part 3 elements and submission due this week)*

- Voltage level translation. Engineering trade-offs. Design reviews. Tolerances & margins.
- Work on final projects. Student/professor meetings. Guest Speaker recording available.

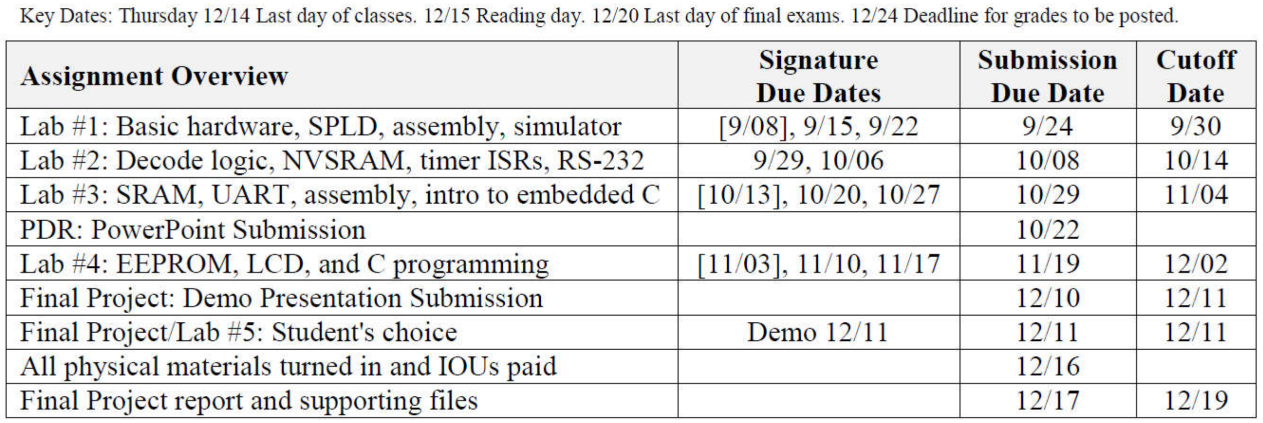
Key Dates of Interest

- 10/30: Week 10 Lecture – Lab #4 topics
- 11/03: Checkpoint for Lab #4 Part #1 Elements
- 11/06: Week 11 Lecture. Work on Lab #4
- 11/10: Signature due date for Lab #4 Part #2 Elements
- 11/13: Week 12 Lecture. Work on Lab #4 and final project
- 11/17: Signature due date for Lab #4 Part #3 Elements
- 11/19: Lab #4 submission due date
- 11/20: Week 13 No Lecture – FALL BREAK
- 11/27: Week 14 Guest Lecture? Work on final projects.
- 12/04: Week 15 Final Lecture
- 12/11: Week 16 Final project demos, last day of classes
- 12/16: All physical materials turned in and IOU's paid
- 12/17: Final project report submissions



December 2023

S	M	T	W	T	F	S
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30



Key Dates: Thursday 12/14 Last day of classes. 12/15 Reading day. 12/20 Last day of final exams. 12/24 Deadline for grades to be posted.

Assignment Overview	Signature Due Dates	Submission Due Date	Cutoff Date
Lab #1: Basic hardware, SPLD, assembly, simulator	[9/08], 9/15, 9/22	9/24	9/30
Lab #2: Decode logic, NVSRAM, timer ISRs, RS-232	9/29, 10/06	10/08	10/14
Lab #3: SRAM, UART, assembly, intro to embedded C	[10/13], 10/20, 10/27	10/29	11/04
PDR: PowerPoint Submission		10/22	
Lab #4: EEPROM, LCD, and C programming	[11/03], 11/10, 11/17	11/19	12/02
Final Project: Demo Presentation Submission		12/10	12/11
Final Project/Lab #5: Student's choice	Demo 12/11	12/11	12/11
All physical materials turned in and IOUs paid		12/16	
Final Project report and supporting files		12/17	12/19

DEBUGGING / GRADING

Debugging

Issue Classification

1. Something is wrong in the hardware design or implementation that is causing wrong/inconsistent behavior
2. Something is wrong in the software design or implementation
3. Something is wrong in the toolset (e.g. SDCC)

Historically, most issues reported in ECEN 5613 have been root caused to categories 1 and 2.

Debugging Strategy

1. Identify the problem
   - Reproduce the problem
   - Understand the expected behavior
2. Isolate the problem
   - Simplify code/hardware to a minimal set
   - Try to identify a light switch test that triggers the problem
3. Understand the error
   - Determine root cause
   - Don’t jump to a conclusion
4. Implement a fix
5. Test and verify
   - Make sure the fix works
   - Make sure the fix doesn’t introduce other issues

Debugging Steps

System checks:

Is it a hardware or software issue?

Hardware checks:

1. Power
2. Resets
3. Clocks
4. Connections

Software checks:

1. Stripped down code experiments
2. Instruction traces (debugging statements, logic analyzer)
3. Interrupt masking during experiments

Debugging Checklist (1/2)

1. All components correctly and fully inserted into WW socket, no bent pins
2. Solid power and ground seen directly at the pins of each chip on the top side of the board (can use multimeter to ohm the connection when the board is powered off).
3. Verify that clocks and reset look correct.
4. Solid solder joints on board. If any question about the quality of the solder joint, the connection can be doubled up by using a wire wrap in addition to the solder/trace connection.
5. All control/data/address signals correctly connected at the NVSRAM. Depending on which 28-pin position of the board is occupied, there are a couple of signals on the board that need to be completed with jumpers. Verify no floating pins.
6. Decoupling capacitors used at every chip location. Bigger decoupling capacitor used on 8051.
7. Verify ALE signal looks clean. Add small capacitor if necessary per the ALE Noise application note. Careful, not too large of a capacitor.

Debugging Checklist (2/2)

8. Write a minimal program (without Paulmon) that simply initializes critical hardware and performs the specific action that shows the problem. Capture the bus transactions with a LogicPort analyzer.
9. Ensure that the LogicPort analyzer thresholds are set to a level that is similar to the threshold of the other chips on the board.
10. Ensure SPLD outputs look good, no glitches during the bus cycle.
11. Have another TA or student help with debug. Sometimes a different set of eyes can see something.
12. Does the student's code work on another board? (if logic is compatible and can support the same instruction sequences)
13. With components removed, is a short (or low impedance) to power or ground seen on the signals in question?
14. If a specific component is suspect, swap that component with a new component

Debugging Example

Situation

- Program utilizes printf() to print a string once. When program is run, the same string is continuously printed on the screen.
- What could be causing this?

Possible Solution

- Simplify code to minimal code that still exhibits this behavior.
- Use logic analyzer to trace the sequence of instructions that are being executed. Use this data to understand the instruction flow.
- Do not allow the main() function to terminate. Put an infinite loop at the end of main(), similar to:

```
main()
{
        …             /* execute regular code here, before the infinite loop */

        while (1);    /* end the main() function with an infinite loop */
}
```

Debugging Example

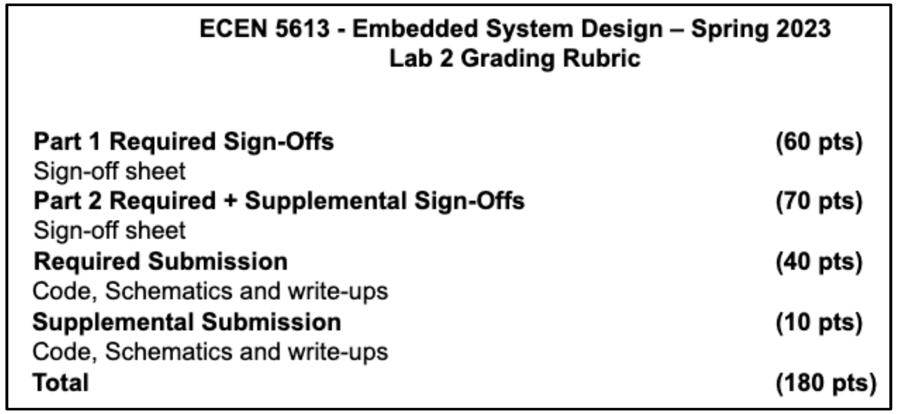
Situation

1. Hardware memory map has 1KB of internal XRAM and 31KB of external XRAM
2. SDCC linker options --xram-loc 0x0000 --xram-size 0x8000
3. Code doesn’t initialize AUXR register

What happens in the following cases?

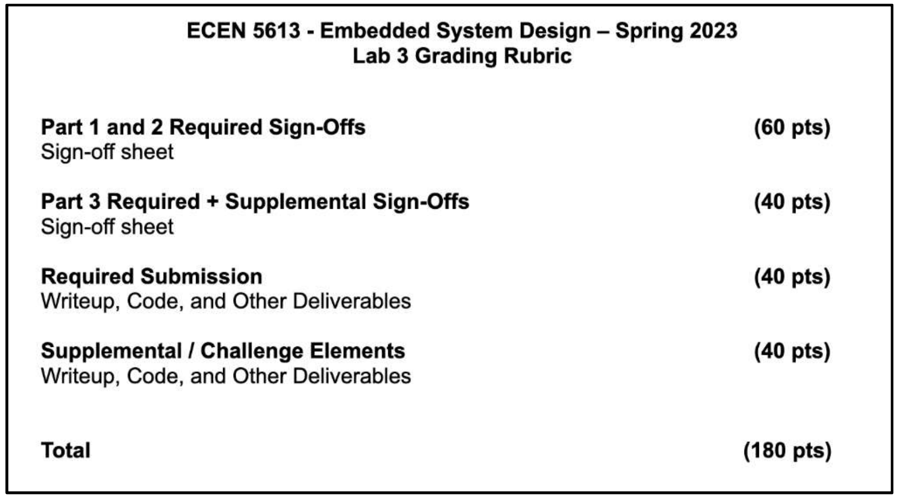
- Case 1: small amount of heap utilization
- Case 2: large amount of heap utilization

Grading



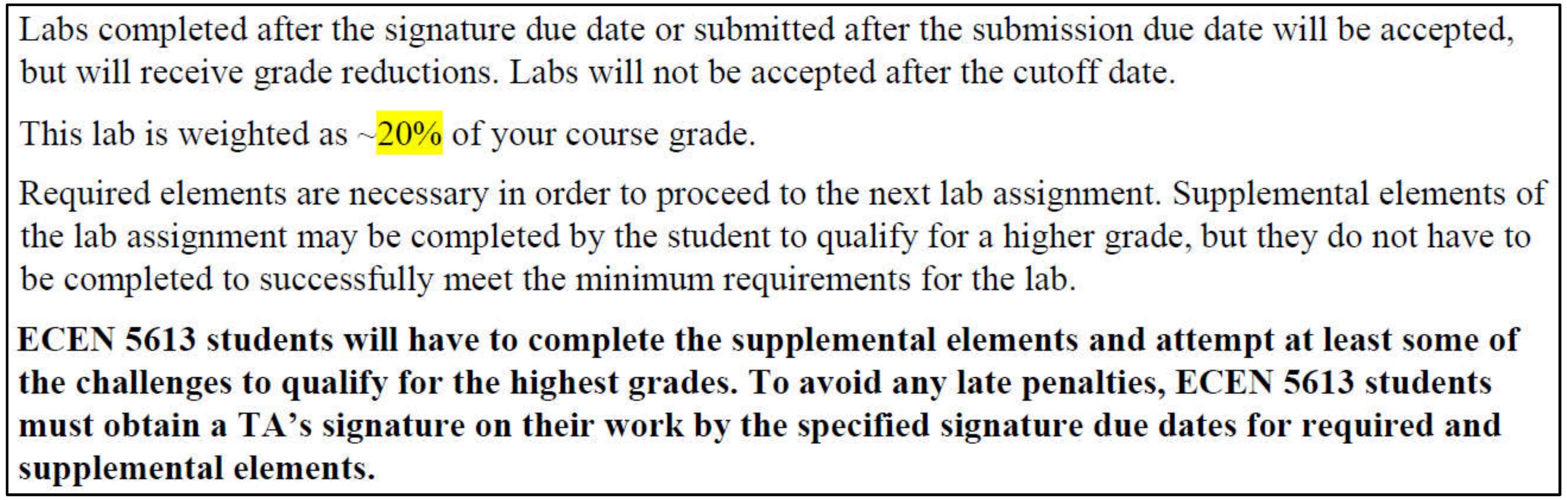
ECEN 5613 - Embedded System Design – Spring 2023
Lab 2 Grading Rubric

Part 1 Required Sign-Offs Sign-off sheet	**(60 pts)**
Part 2 Required + Supplemental Sign-Offs Sign-off sheet	**(70 pts)**
Required Submission Code, Schematics and write-ups	**(40 pts)**
Supplemental Submission Code, Schematics and write-ups	**(10 pts)**
Total	**(180 pts)**



ECEN 5613 - Embedded System Design – Spring 2023
Lab 3 Grading Rubric

Part 1 and 2 Required Sign-Offs Sign-off sheet	**(60 pts)**
Part 3 Required + Supplemental Sign-Offs Sign-off sheet	**(40 pts)**
Required Submission Writeup, Code, and Other Deliverables	**(40 pts)**
Supplemental / Challenge Elements Writeup, Code, and Other Deliverables	**(40 pts)**
Total	**(180 pts)**

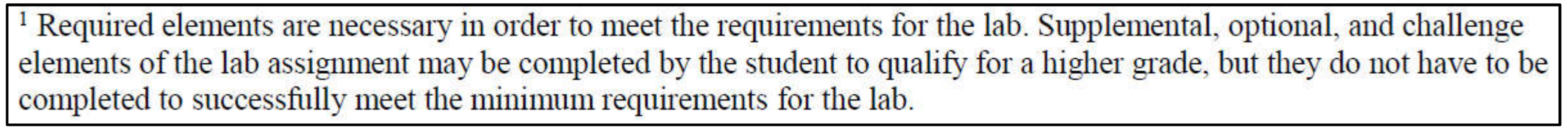


Labs completed after the signature due date or submitted after the submission due date will be accepted, but will receive grade reductions. Labs will not be accepted after the cutoff date.

This lab is weighted as ~20% of your course grade.

Required elements are necessary in order to proceed to the next lab assignment. Supplemental elements of the lab assignment may be completed by the student to qualify for a higher grade, but they do not have to be completed to successfully meet the minimum requirements for the lab.

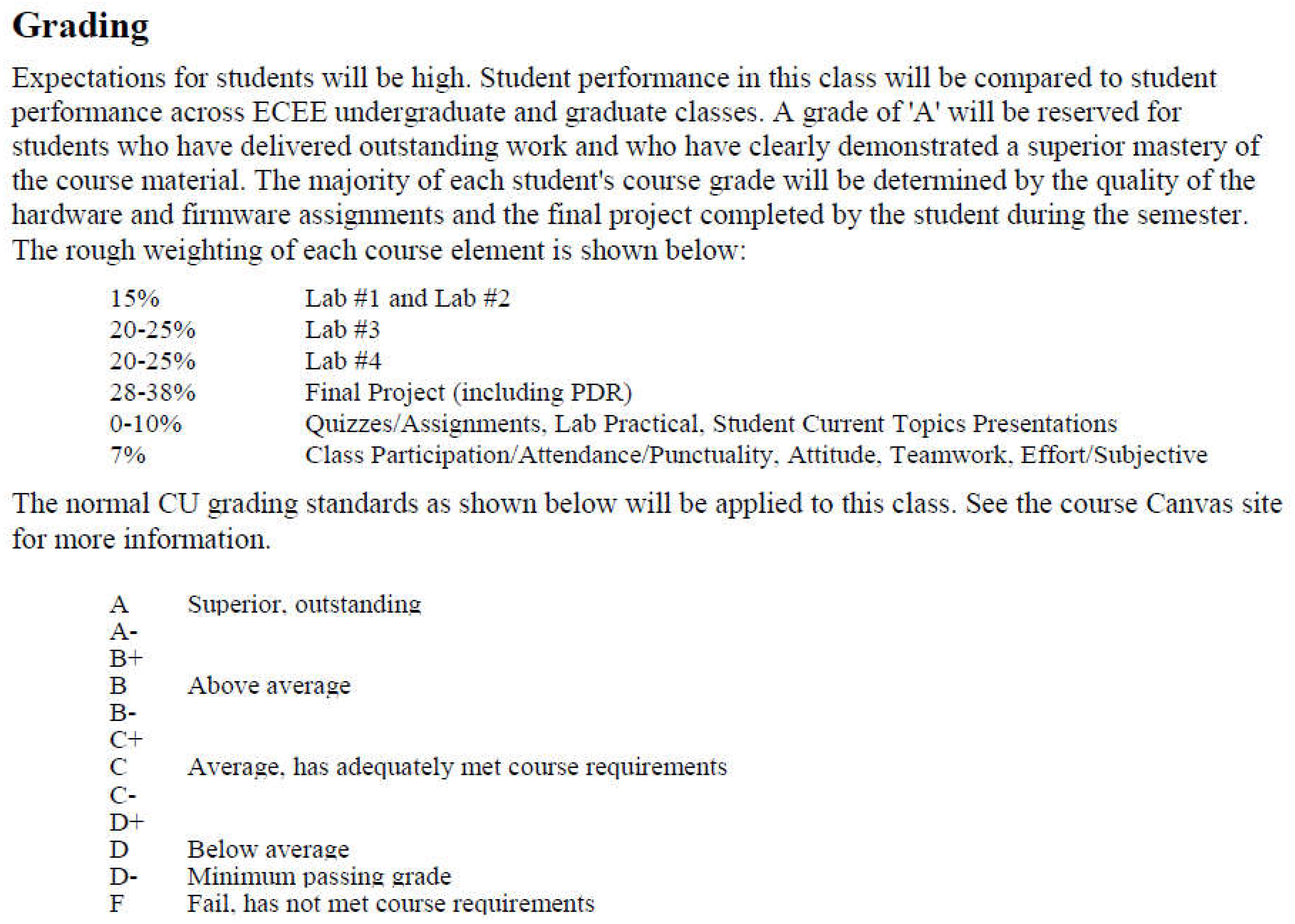
ECEN 5613 students will have to complete the supplemental elements and attempt at least some of the challenges to qualify for the highest grades. To avoid any late penalties, ECEN 5613 students must obtain a TA's signature on their work by the specified signature due dates for required and supplemental elements.



[1] Required elements are necessary in order to meet the requirements for the lab. Supplemental, optional, and challenge elements of the lab assignment may be completed by the student to qualify for a higher grade, but they do not have to be completed to successfully meet the minimum requirements for the lab.

Q: If I complete only the required elements, does that qualify me for the highest grade, which is an 'A'?

A: No, the required elements are considered the minimum requirements for each lab. Students that complete additional elements will qualify for the higher grades awarded in this course.



Grading

Expectations for students will be high. Student performance in this class will be compared to student performance across ECEE undergraduate and graduate classes. A grade of 'A' will be reserved for students who have delivered outstanding work and who have clearly demonstrated a superior mastery of the course material. The majority of each student's course grade will be determined by the quality of the hardware and firmware assignments and the final project completed by the student during the semester. The rough weighting of each course element is shown below:

15%	Lab #1 and Lab #2
20-25%	Lab #3
20-25%	Lab #4
28-38%	Final Project (including PDR)
0-10%	Quizzes/Assignments, Lab Practical, Student Current Topics Presentations
7%	Class Participation/Attendance/Punctuality, Attitude, Teamwork, Effort/Subjective

The normal CU grading standards as shown below will be applied to this class. See the course Canvas site for more information.

A	Superior, outstanding
A-	
B+	
B	Above average
B-	
C+	
C	Average, has adequately met course requirements
C-	
D+	
D	Below average
D-	Minimum passing grade
F	Fail, has not met course requirements

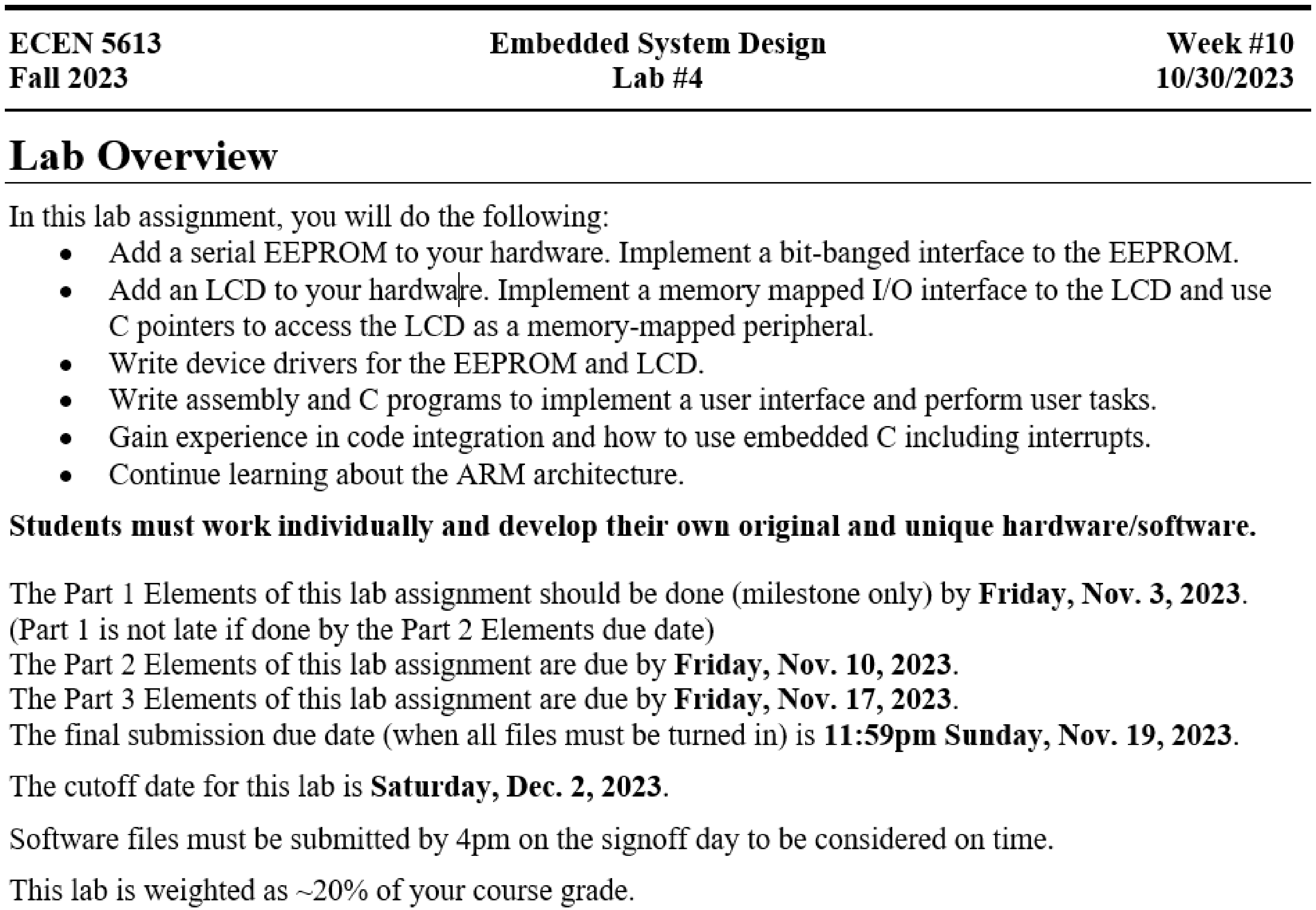
Source: ECEN 5613 Syllabus

Class Participation Grade

Some students have asked what factors can be considered in the class participation part of the course grade. Here are some suggestions:

- Attendance during lecture, being present and actively listening/engaging in learning.
- Having camera on during lecture, during presentations, and in 1:1 meetings.
- Completing informal surveys/evaluations for the course.
- Engaging in appropriate class discussions.
- Staying current on reading Slack and e-mail messages for this course.
- Using appropriate reactions (e.g. thumbs up) in Zoom and Slack to show that you read or understood comments that have been made.
- Engaging in positive interaction with the TA's and instructor.
- Being helpful to TA's and classmates.

Lab #4 Overview



ECEN 5613 | **Embedded System Design** | **Week #10**
Fall 2023 | **Lab #4** | **10/30/2023**

Lab Overview

In this lab assignment, you will do the following:

- Add a serial EEPROM to your hardware. Implement a bit-banged interface to the EEPROM.
- Add an LCD to your hardware. Implement a memory mapped I/O interface to the LCD and use C pointers to access the LCD as a memory-mapped peripheral.
- Write device drivers for the EEPROM and LCD.
- Write assembly and C programs to implement a user interface and perform user tasks.
- Gain experience in code integration and how to use embedded C including interrupts.
- Continue learning about the ARM architecture.

Students must work individually and develop their own original and unique hardware/software.

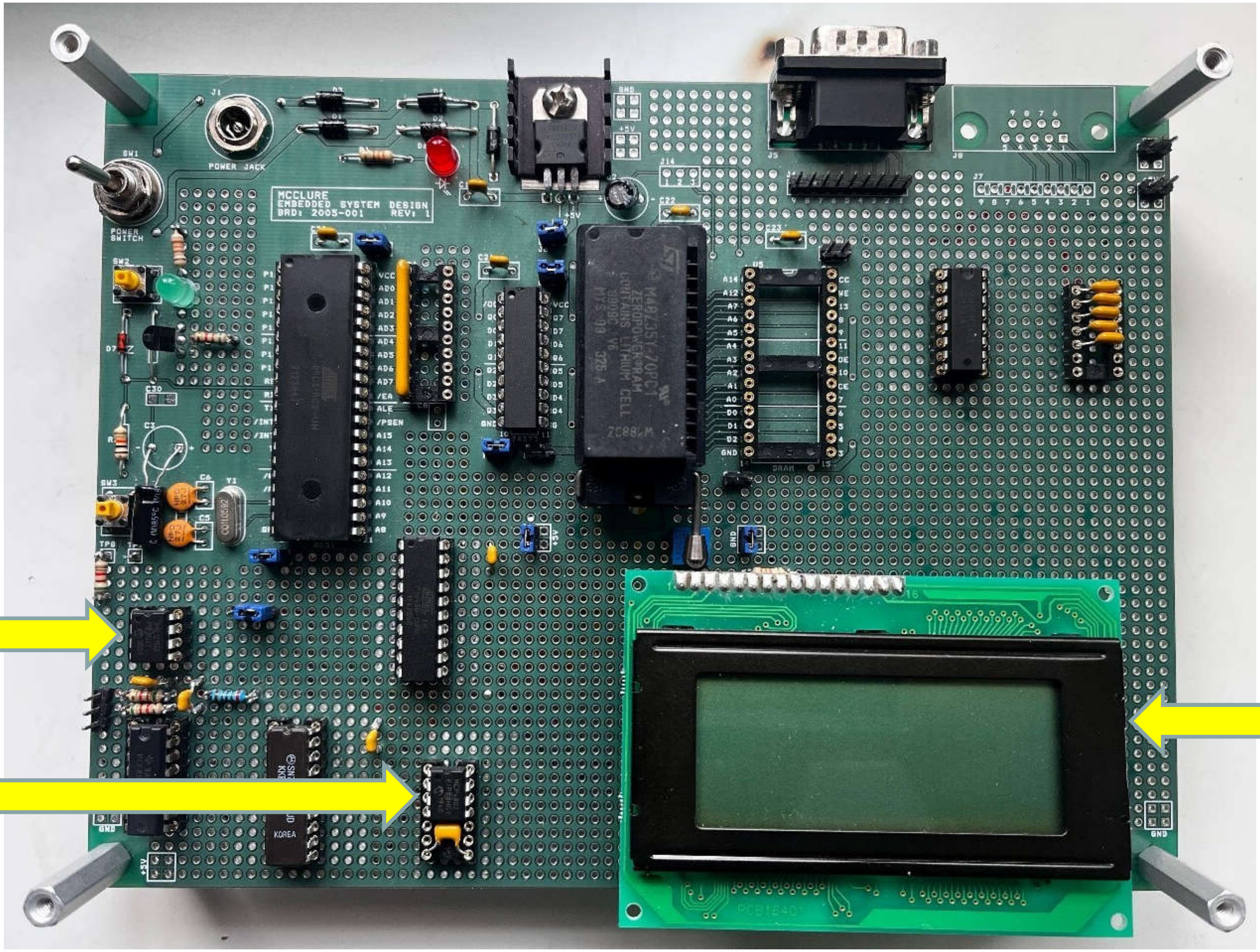
The Part 1 Elements of this lab assignment should be done (milestone only) by **Friday, Nov. 3, 2023**.
(Part 1 is not late if done by the Part 2 Elements due date)
The Part 2 Elements of this lab assignment are due by **Friday, Nov. 10, 2023**.
The Part 3 Elements of this lab assignment are due by **Friday, Nov. 17, 2023**.
The final submission due date (when all files must be turned in) is **11:59pm Sunday, Nov. 19, 2023**.

The cutoff date for this lab is **Saturday, Dec. 2, 2023**.

Software files must be submitted by 4pm on the signoff day to be considered on time.

This lab is weighted as ~20% of your course grade.

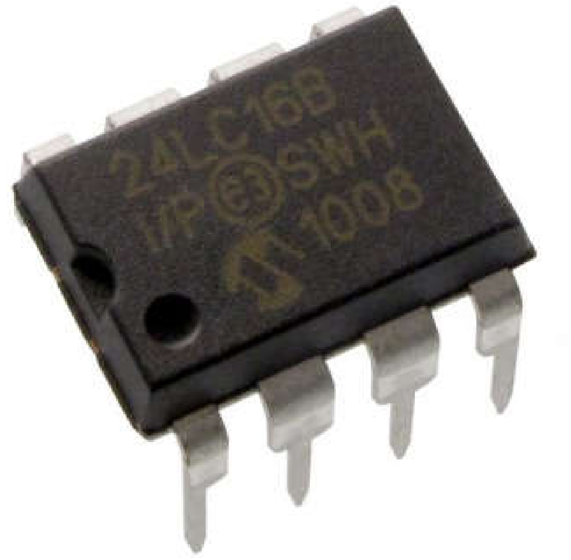
Lab #4 Overview



I^{2}C EEPROM

I^{2}C EEPROM

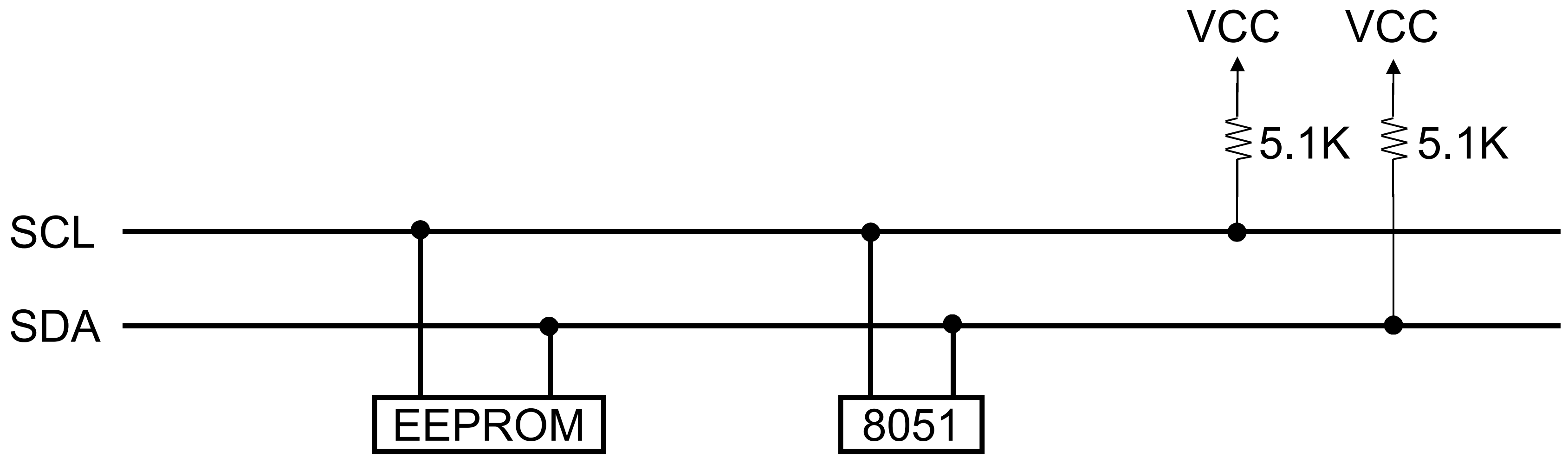
- Why EEPROM?
  - Non-volatile memory
  - In-circuit programming/erasing
  - Inexpensive
- Applications
  - Speed dial on phone
  - Retain settings on car, TV, other devices
- Device contains a tunneling dielectric
- Erased state is when the bit cell is storing a ‘1’
- ~20V is generated by internal charge pump in order to program the device



I^2C Overview

- I^2C = IIC = Inter-Integrated Circuit
- Synchronous bi-directional protocol
- 2-wire interface
  - SCL – Clock
  - SDA – Data
- With standard I^2C, up to 16kbits (2KB) of EEPROM maximum on a single I^2C bus
- Supports devices such as EEPROM and data converters
- I^2C protocol from 1980's
  - Standard (100 kbps)
  - Fast (400 kbps)
  - High-Speed (3.4 Mbps)
  - Ultra-Fast (5 Mbps unidirectional)
- Other related serial interfaces
  - SPI, Microwire, 3-wire, 1-wire, SMBus, CANBUS, I3C (Improved IIC)

8051 Connections



SCL pull-up somewhat optional in our implementation since 8051 is the only master. If more than one master, must use open drain.

Reduce pull-up value if using a Fast-mode I^{2}C device.

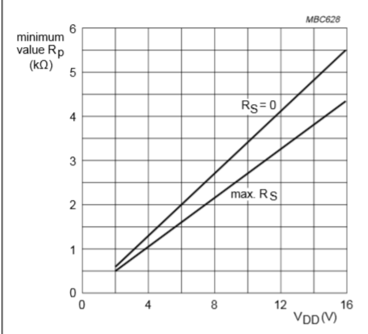


Fig.37 Minimum value of R_p as a function of supply voltage with the value os R_s as a parameter.

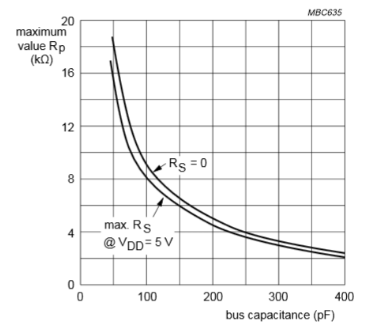


Fig.39 Maximum value of R_p as a function of bus capacitance for a Standard-mode I^{2}C-bus.

Source: Philips (NXP) I^{2}C Bus Specification

Implementation

- Hardware
  - Use decoupling capacitor
  - Use pull-up resistors for SCL and SDA
  - Pick two port pins on the 8051
- Firmware
  - Check generation of SCL with oscilloscope, ensure it doesn’t exceed maximum bus speed (e.g. 100 kHz)
  - I^2C functions for SDCC available on web
  - Can use ACK polling to determine device readiness

I^2C Terms

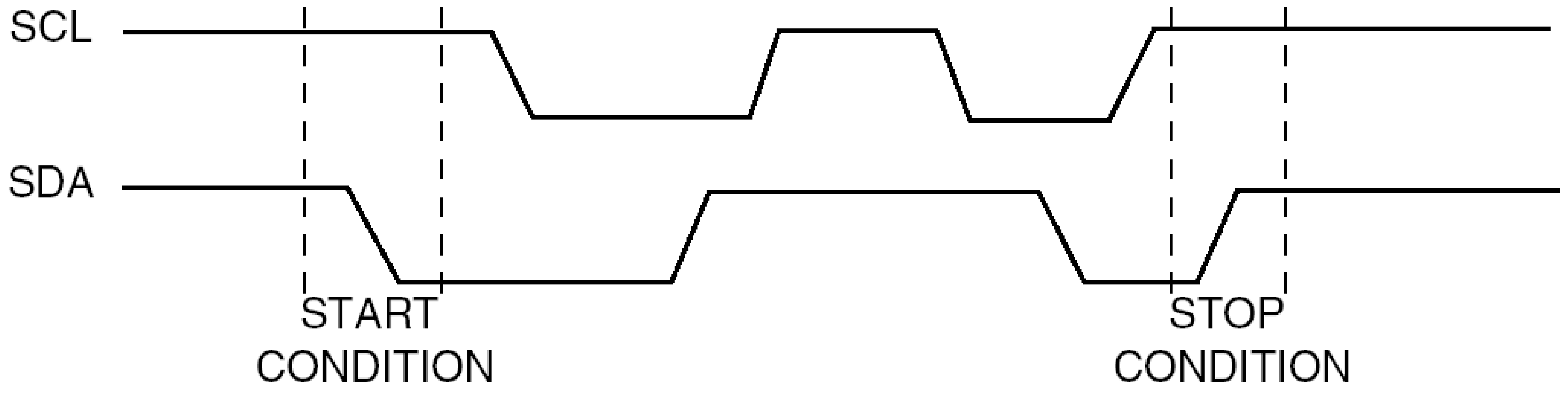
- Master (8051)
  - Device which initiates all activity on the bus
  - Generates clock for the transfer
  - Generates start and stop conditions
- Subordinate (Slave) (EEPROM)
  - Responds to the master’s request
- Receiver
  - Reads data on the bus. Can be Master or Subordinate.
- Transmitter
  - Writes data to the bus. Can be Master or Subordinate.
- Page Block
  - 2Kbits of data → 256 bytes → 16 pages
- Page
  - 16 bytes (8 byte pages for small EEPROMs like 24C02)

I^{2}C Protocol – Bus Conditions

Four bus conditions

- Start
  - High to low transition of SDA when SCL is high. Generated by Master only.
- Stop
  - Low to high transition of SDA when SCL is high. Generated by Master only.
- Data
  - Data must be valid when SCL is high
  - Data can change when SCL is low
- ACK
  - Receiver drives this condition

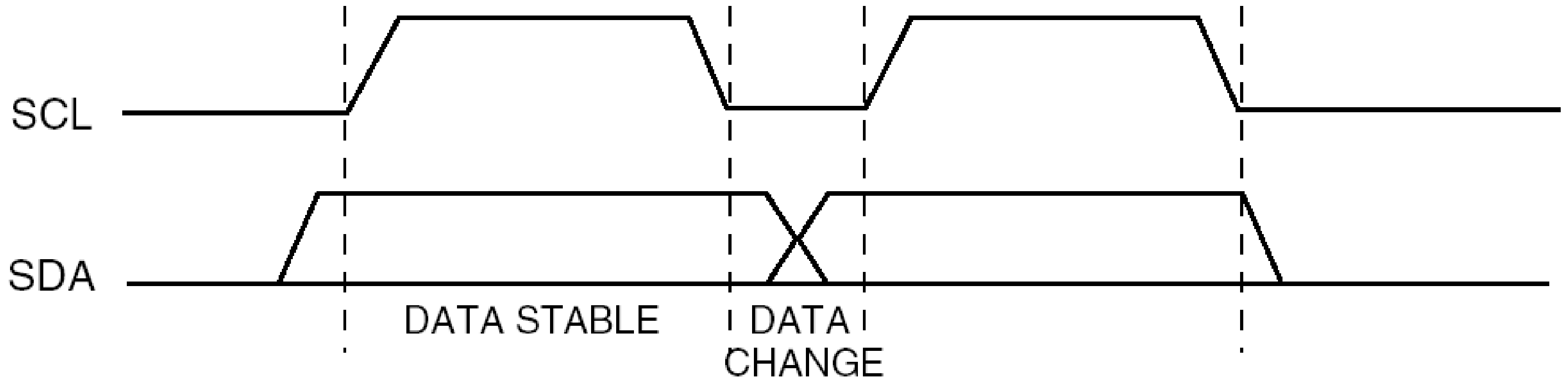
Start and Stop Definition



- Start
  - High to low transition of SDA when SCL is high. Generated by Master only.
- Stop
  - Low to high transition of SDA when SCL is high. Generated by Master only.

Source: Fairchild/National NM24C16/17 Datasheet

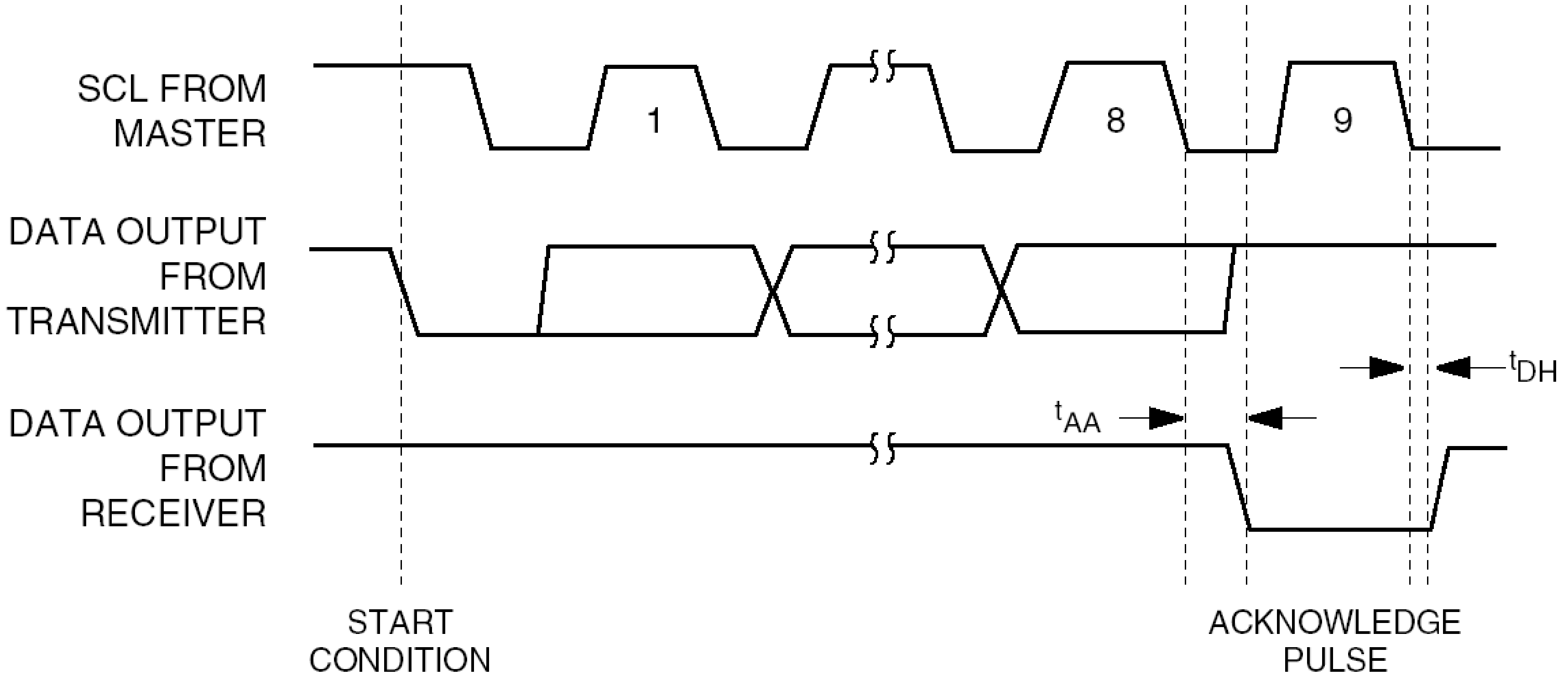
Data Validity



Data must be valid when SCL is high
Data can change when SCL is low

Source: Fairchild/National NM24C16/17 Datasheet

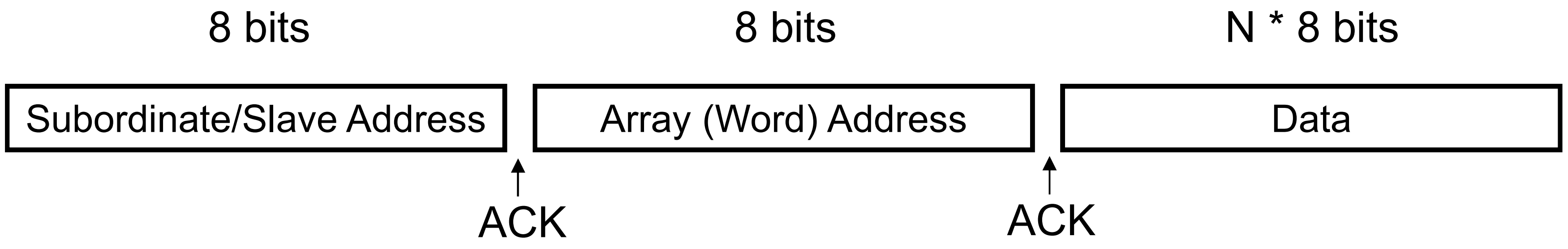
Acknowledge Response from Receiver

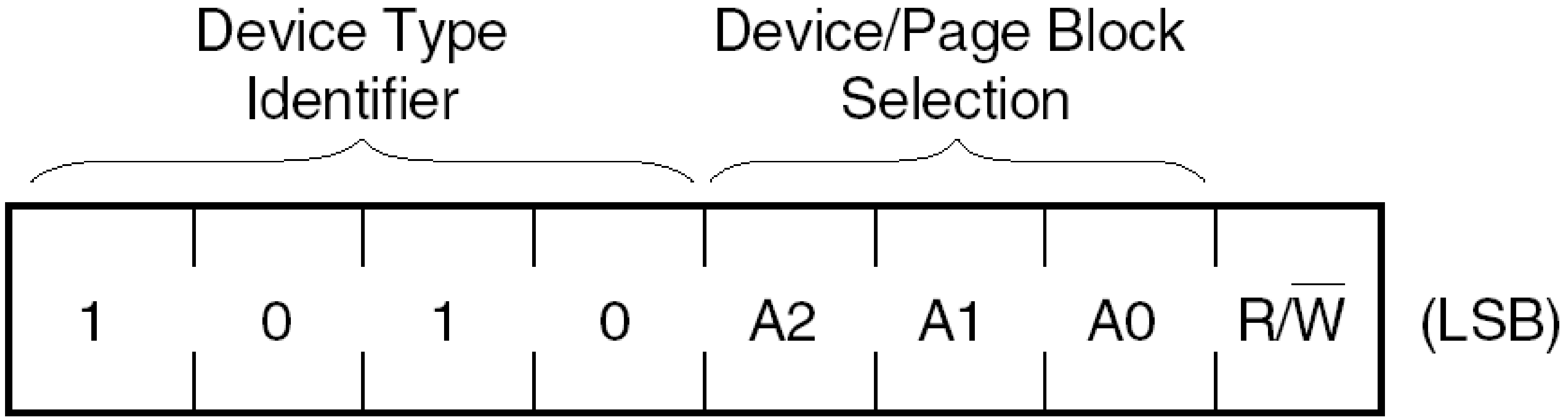


- Lack of an acknowledge from the receiver can indicate an error or that the receiver is busy

Source: Fairchild/National NM24C16/17 Datasheet

I^2C Protocol – Addressing

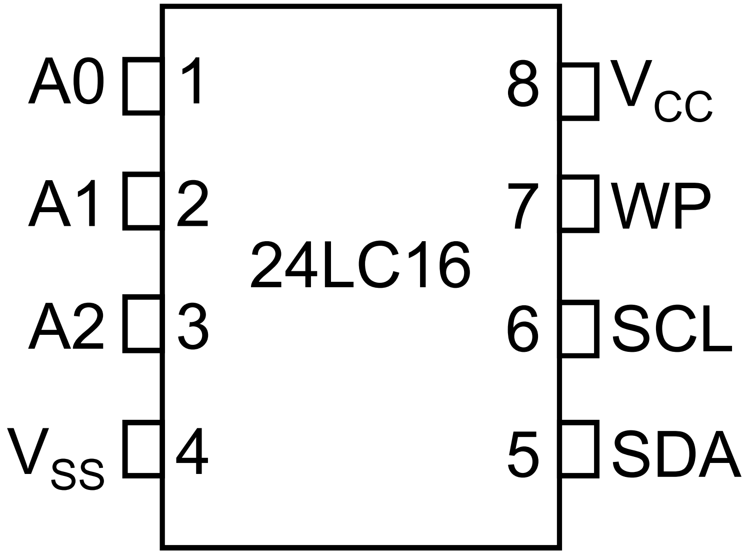




Subordinate/Slave Address

Source: Fairchild/National NM24C16/17 Datasheet

EEPROM Page Block Selection



24XX16: 2KB (8 page blocks)

24XX08: 1KB (4 page blocks)

24XX04: 512 bytes (2 page blocks)

24XX02: 256 bytes

24XX01: 128 bytes

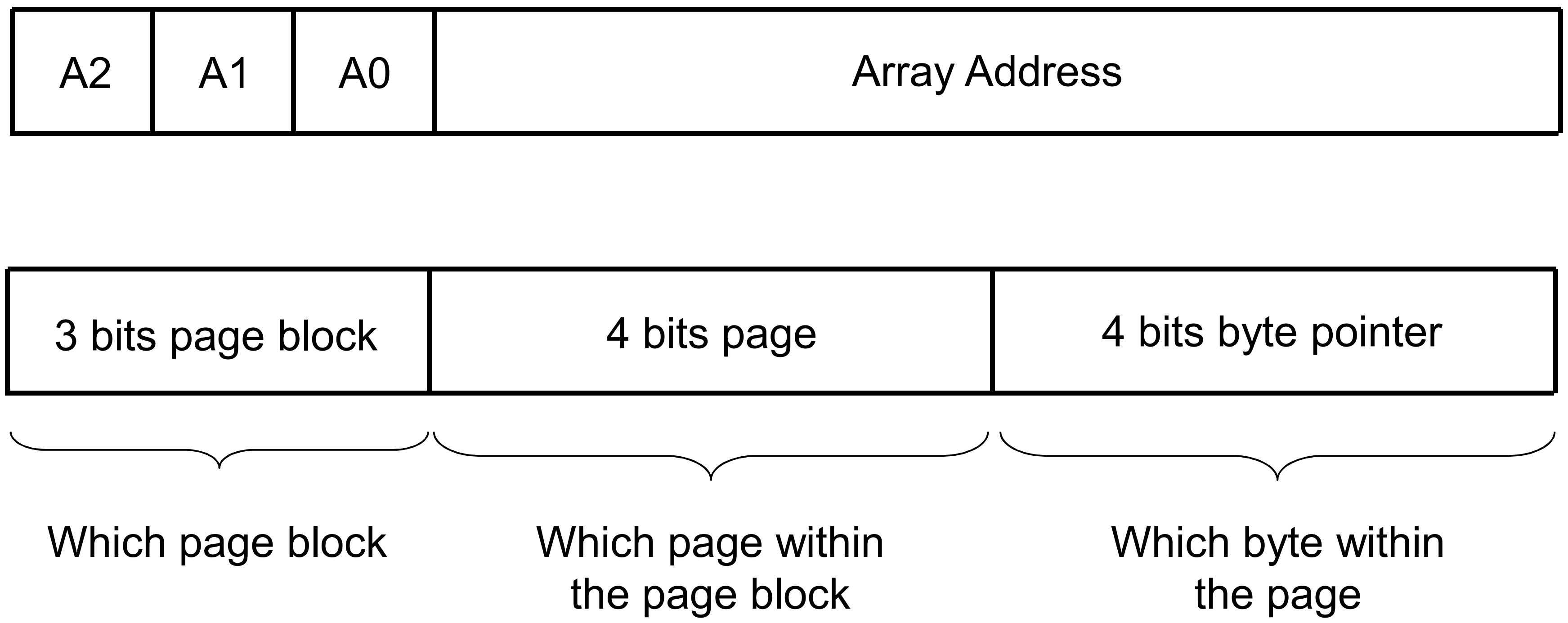
24XX00: 16 bytes

24XX16: A2, A1, and A0 are NC (no-connect). Page block is selected using 3 address bits in subordinate/slave address field of command.

24XX04: A0 is NC (A2:A1 pins identify one of four devices possible on the I^{2}C bus). Device is selected using most significant 2 address bits in subordinate/slave address byte (these must match the strapping of A2 and A1 pins on the chip). Page block is selected using A0 address bit in subordinate/slave address field of command.

EEPROM Internal Address Pointer

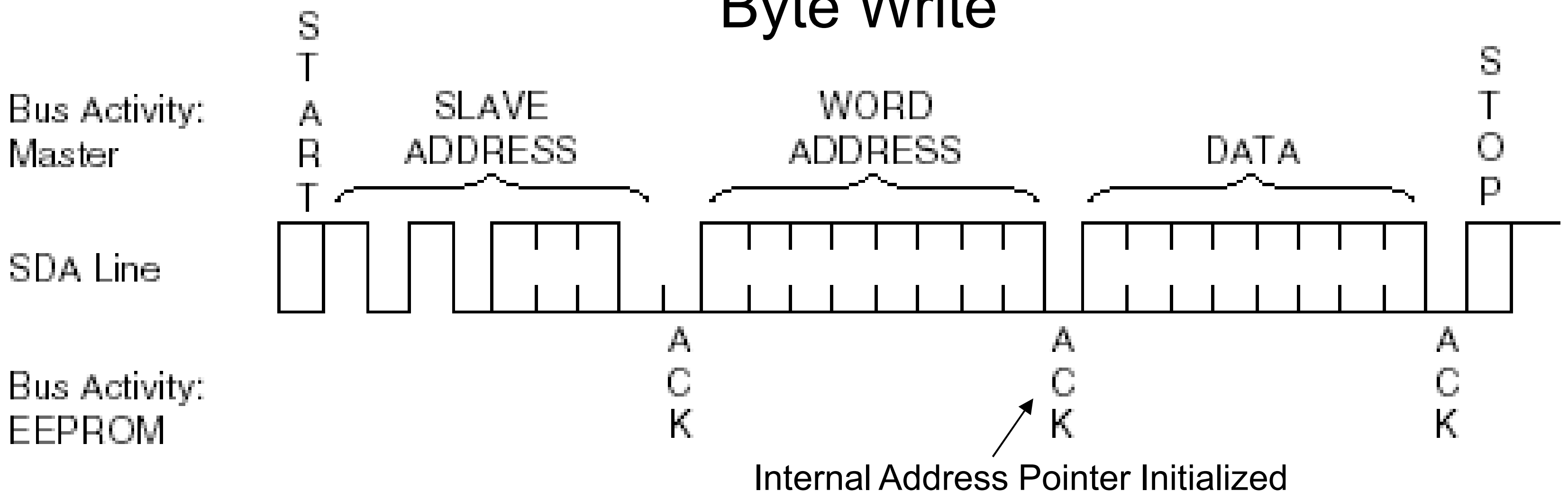
For the 24XX16, an 11-bit internal address pointer uniquely identifies the addressed memory cell within the EEPROM



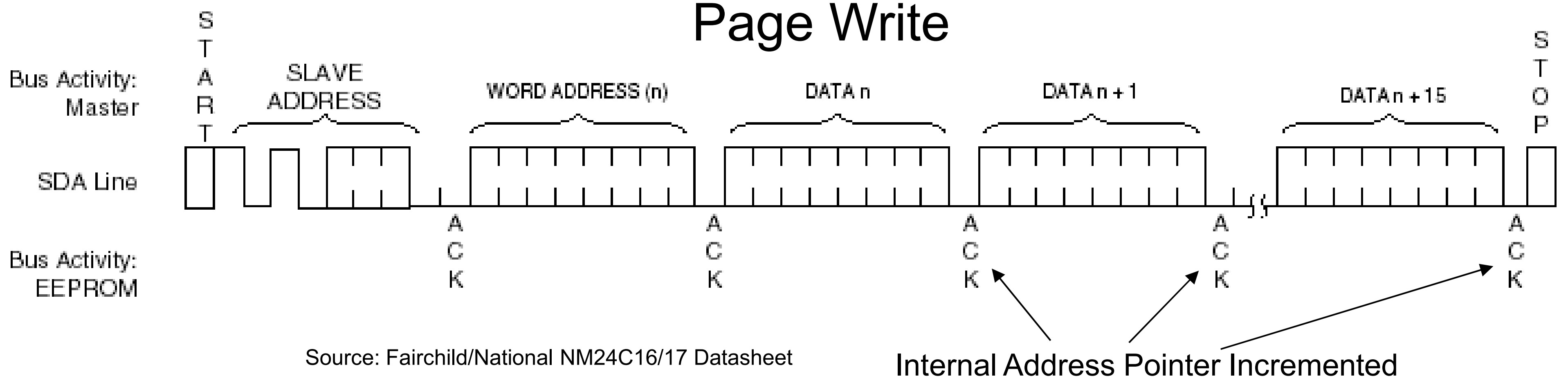
(these four bits associated with circular volatile buffer in EEPROM)

Write Operations

Byte Write

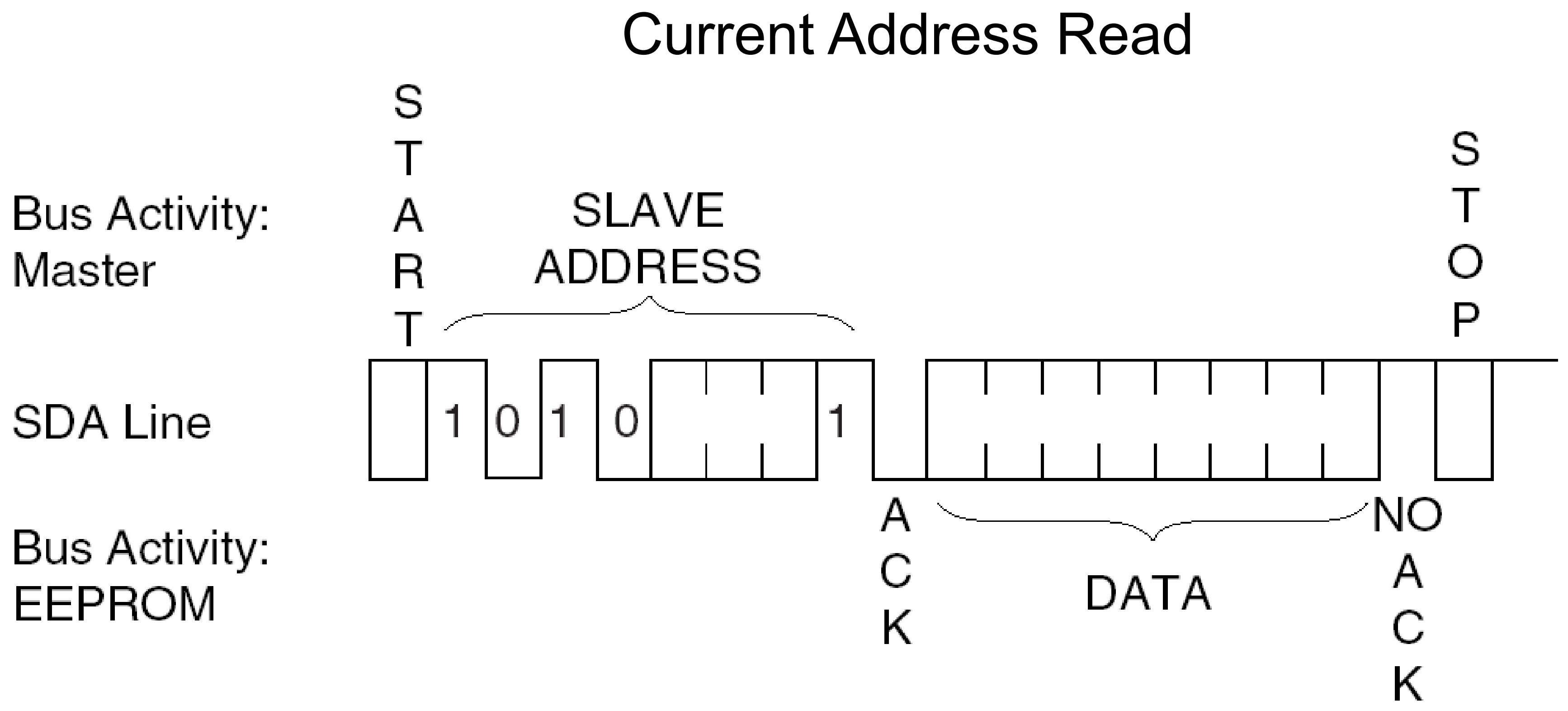


Page Write



Source: Fairchild/National NM24C16/17 Datasheet

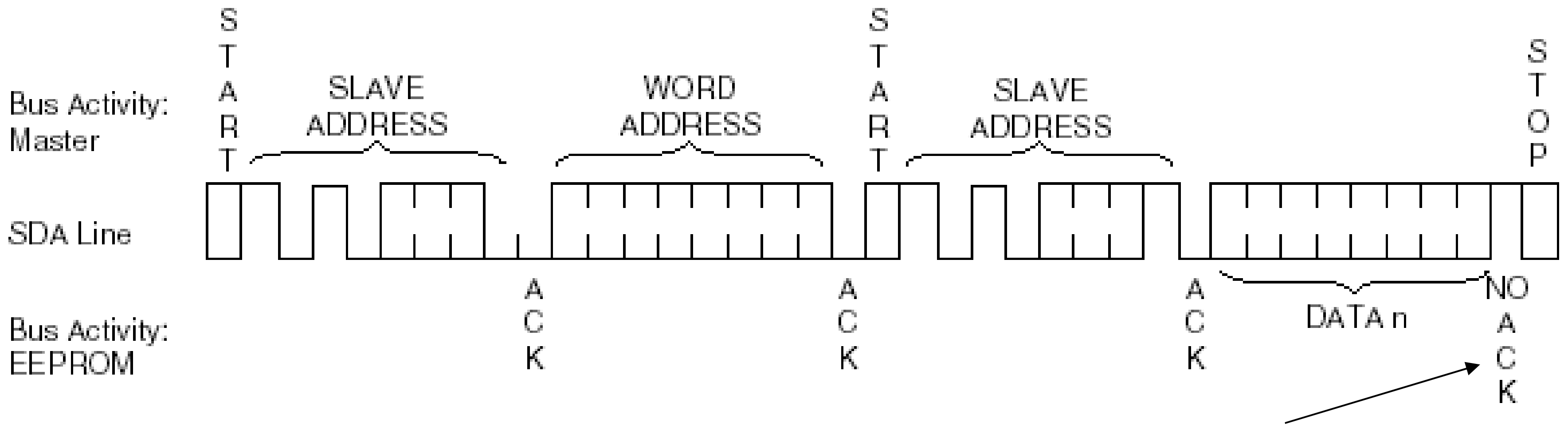
Read Operations



Source: Fairchild/National NM24C16/17 Datasheet

Read Operations

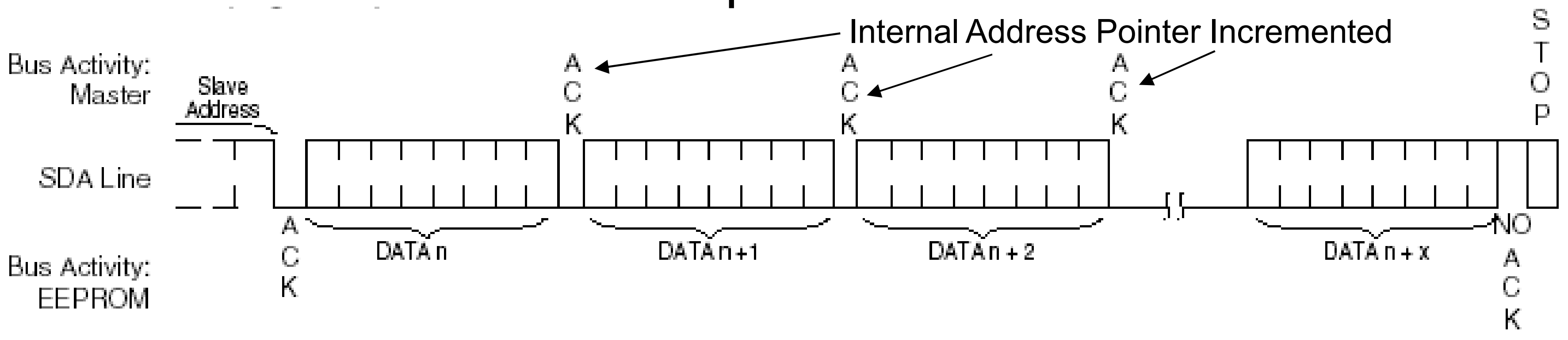
Random Read



Internal Address Pointer Not Incremented

Sequential Read

Internal Address Pointer Incremented



Source: Fairchild/National NM24C16/17 Datasheet

I2C Bit Banging

```
*----------------------------
* I2C BUSS Interface routines
*----------------------------
*
* Generate a START condition & TX char in A
*
?ICstrt SETB    P3.4        SDA = 1
        SETB    P1.0        SCL = 1
        NOP
        CLR     P3.4        SDA = 0 - START
        NOP
        CLR     P1.0        SCL = 0 - Ready for first bit
*
* Send 8 bit character in A
*
?ICsend MOV     R7,#8       8 bits per byte
?1      RLC     A           Get next bit to send
        MOV     P3.4,C      Write bit to data line
        SETB    P1.0        Toggle CLOCK high
        NOP                 Waste time
        CLR     P1.0        Toggle CLOCK low
        DJNZ    R7,?1       Send all bits
        SETB    P3.4        Release DATA line
        SETB    P1.0        9'th clock
        NOP
        MOV     C,P3.4      Get ACK bit
        CLR     P1.0        Return clock LOW
        RET
```

```
*
* Read 8 bits of data into ACC
*
?ICread MOV     R7,#8       8 bits per byte
?2      SETB    P1.0        Set clock high
        NOP
        MOV     C,P3.4      Get bit
        RLC     A           Shift it over
        CLR     P1.0        Reset bit
        DJNZ    R7,?2       Do them all
        RET
*
* Send an ACK to the remote
*
?ICack  CLR     P3.4        Zero DATA
        SETB    P1.0        Raise CLOCK
        NOP
        CLR     P1.0        Lower CLOCK
        SETB    P3.4        Release DATA
        RET
```

Note: Other I^2C libraries for SDCC are available on the Internet

Source: Dunfield Development Systems MICRO-C I2C.ASM

Other

- Wear leveling
- Over-provisioning, spare blocks/sectors
- Write amplification
- Data buffering/battery backup

Learning Check

1. What is the benefit of the page write feature?

2. Consider the following:
   - Assume initial state of EEPROM cells 0x00-0xFF is 0xFF

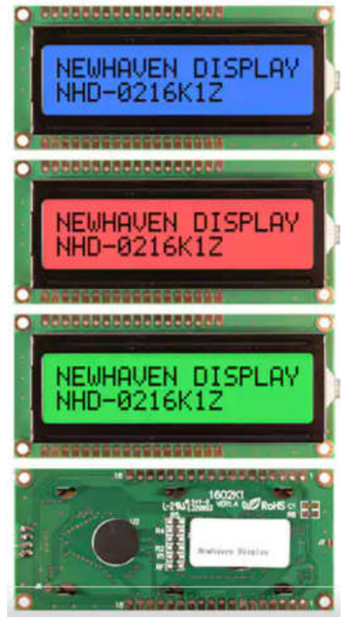
0x 00	0x 01	0x 02	0x 03	0x 04	0x 05	0x 06	0x 07	0x 08	0x 09	0x 0A	0x 0B	0x 0C	0x 0D	0x 0E	0x 0F	0x 10	0x 11	0x 12
0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF	0x FF

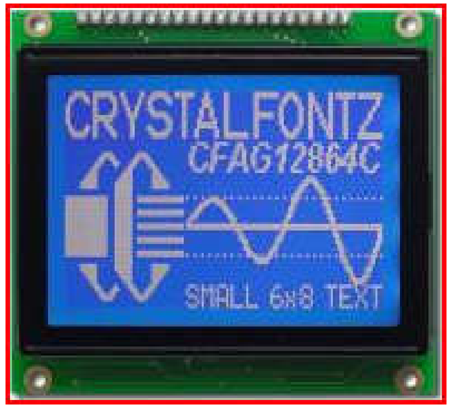
   - Now perform page write at address 0x09 with data bytes 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09.
   - What does the EEPROM contain after the page write?

LCDs
Liquid Crystal Displays

LCD Types and Connectors

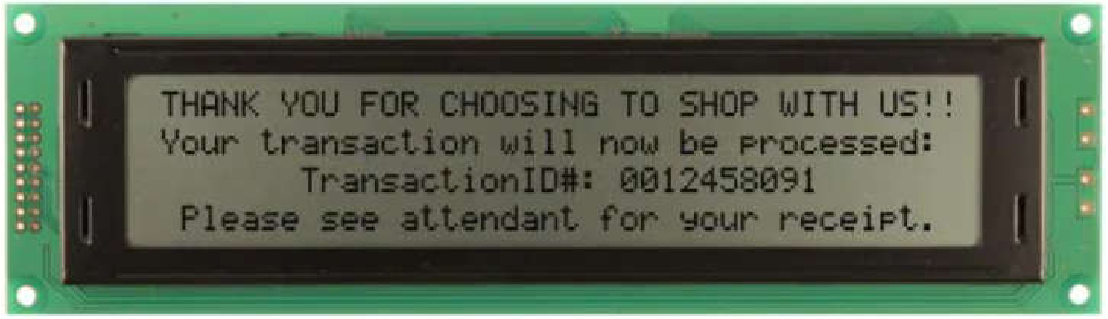
Graphic and Alphanumeric/Character/Text





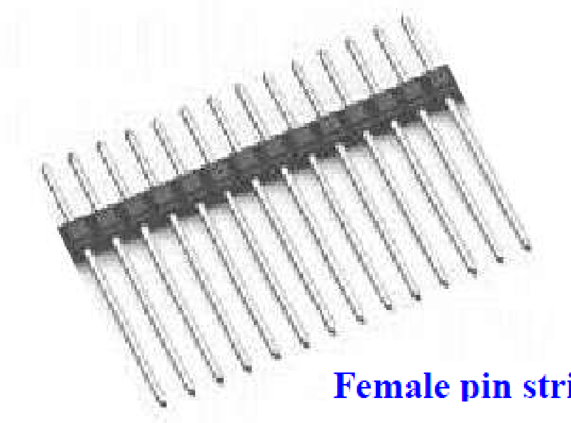
Graphic LCDs with Backlight

14-16 pin interface

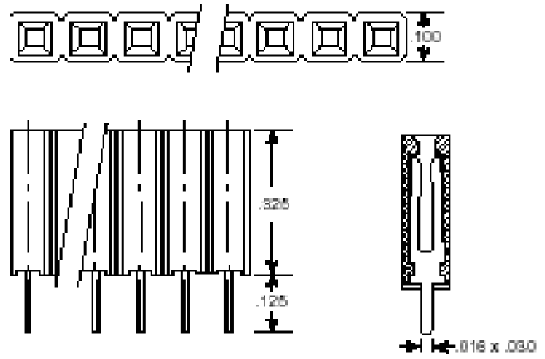

THANK YOU FOR CHOOSING TO SHOP WITH US!!
Your transaction will now be processed:
TransactionID#: 0012458091
Please see attendant for your receipt.

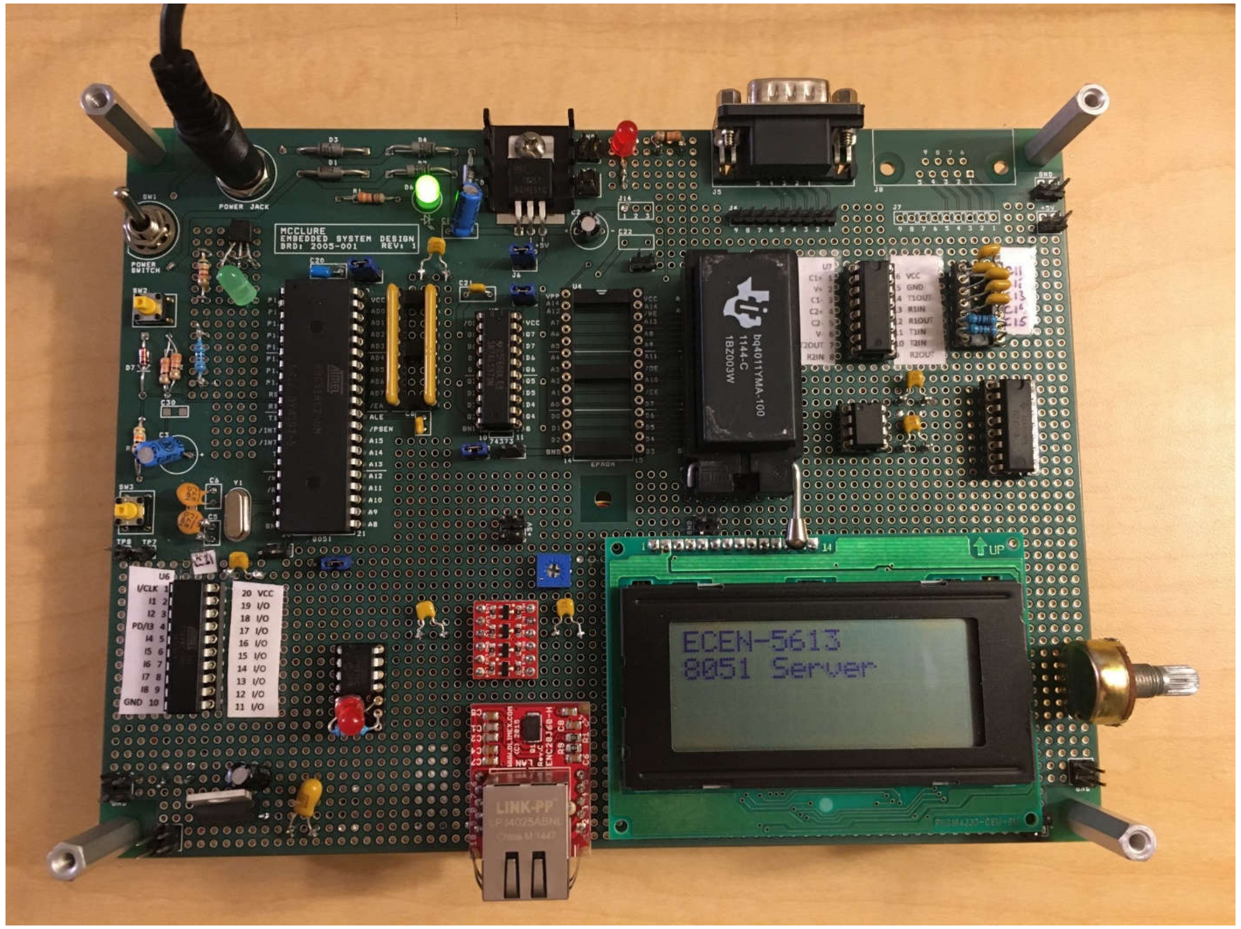
Text/Alphanumeric LCDs with No Backlight

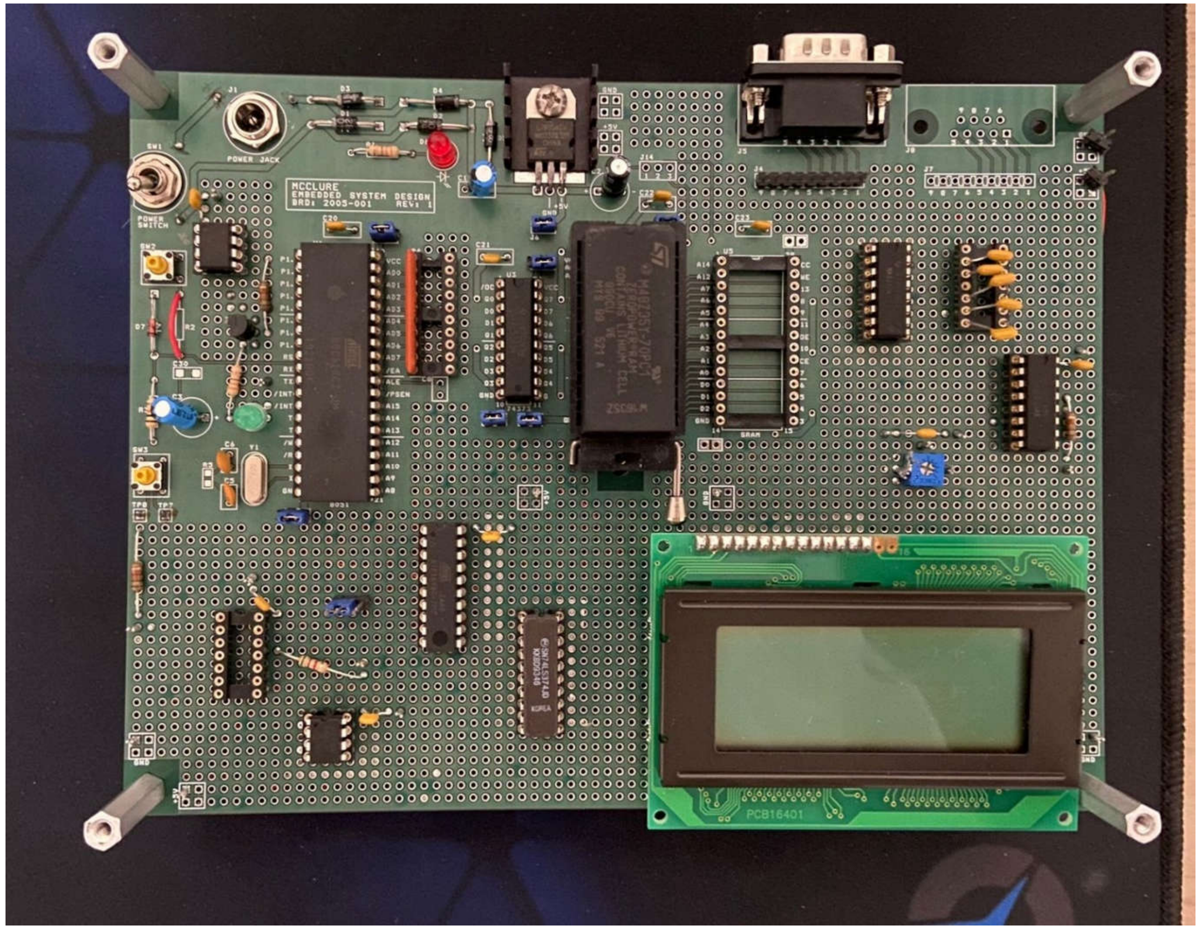
Male pin strip header

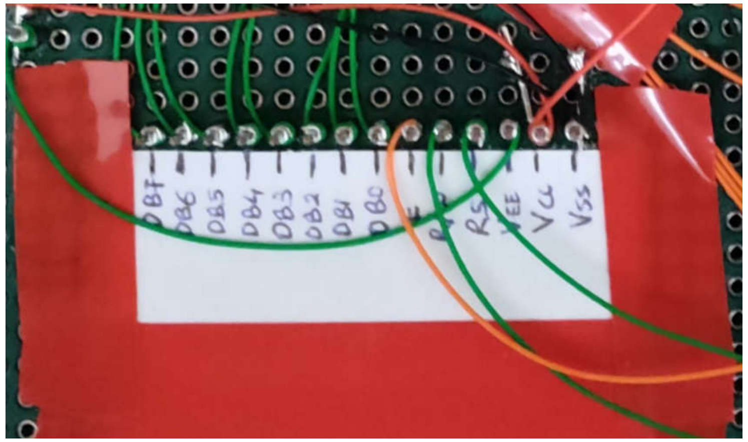


Female pin strip socket




ECEN-5613
8051 Server




DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
E
R/W
Rs
VEE
Vcc
Vss

LCD Lecture Notes

Each LCD display location where a character can be displayed has a unique address in display data RAM (DDRAM). To display a character on the LCD, write the character code to the appropriate address in DDRAM. The DDRAM addresses for the DMC20434 (20x4) LCD are mapped to the LCD panel as in Figure 1. The DDRAM addresses for the DMC16433 (16x4) LCD are similar to the DMC20434.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Figure 1: 20x4 LCD Addresses (given in hexadecimal)

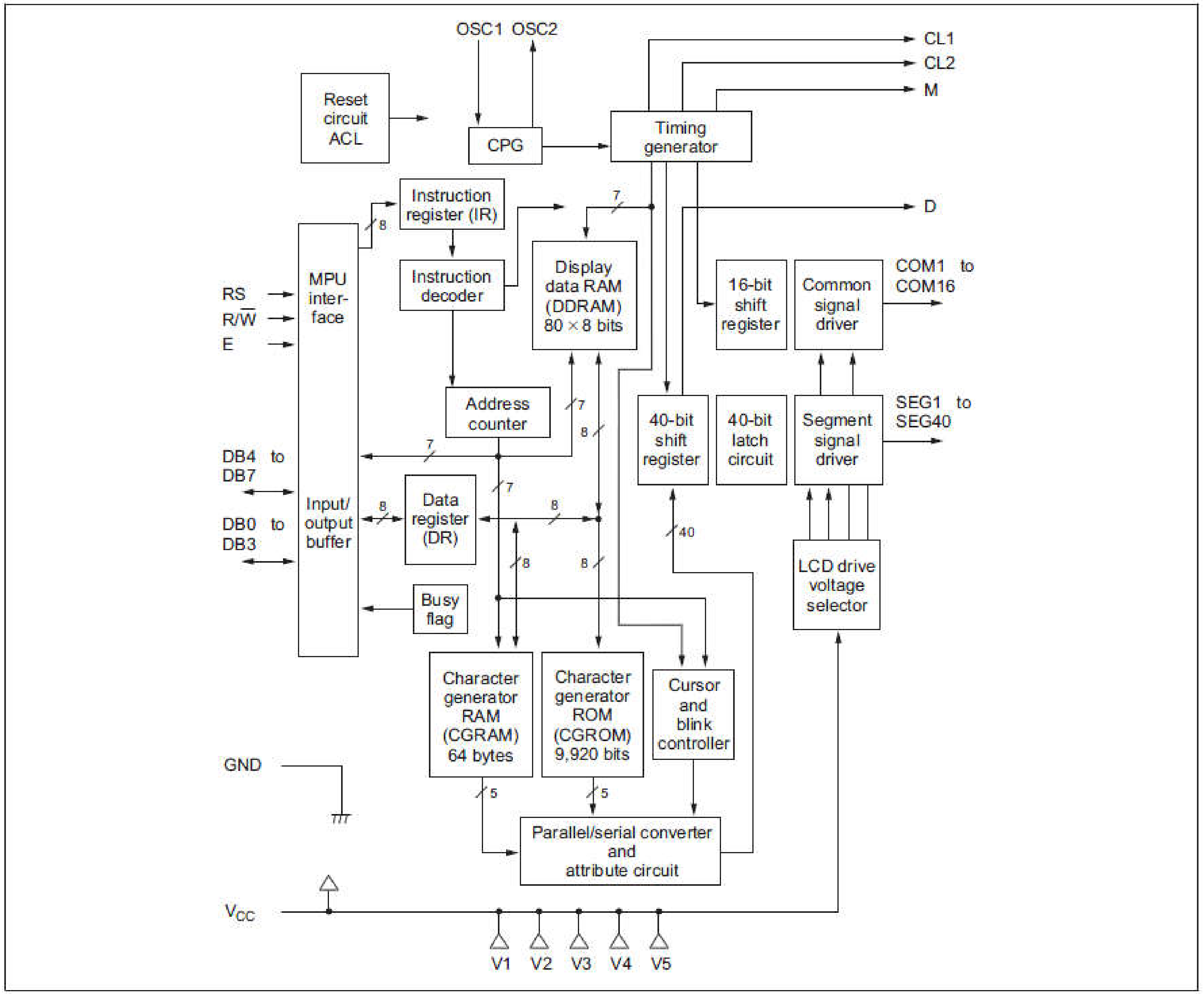
The DDRAM addresses for the 16x1 LCD are mapped to the LCD panel as follows:

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F

Figure 2: 16x1 LCD Addresses (given in hexadecimal)

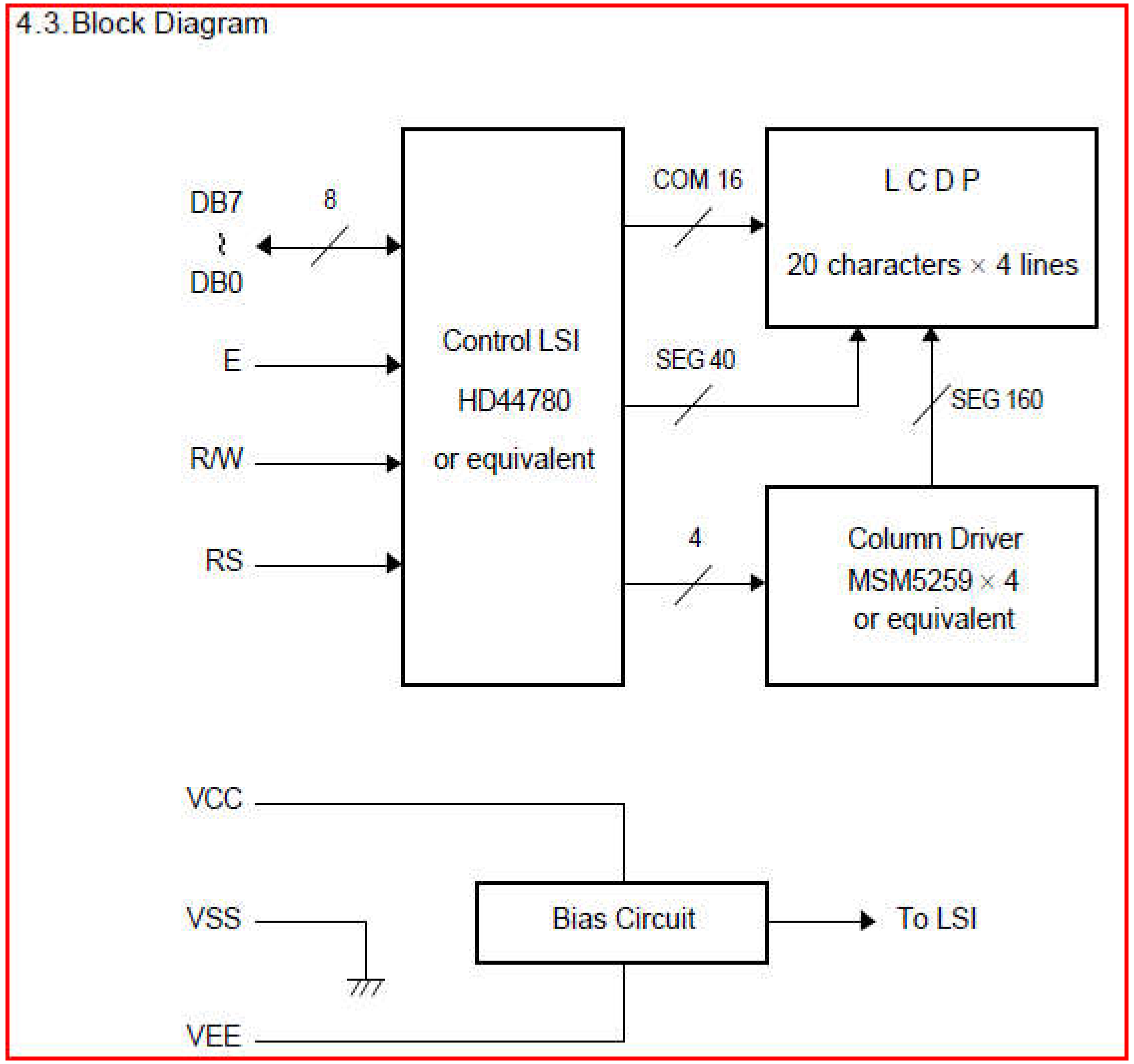
Source: Course Lecture Notes

HD44780U Block Diagram



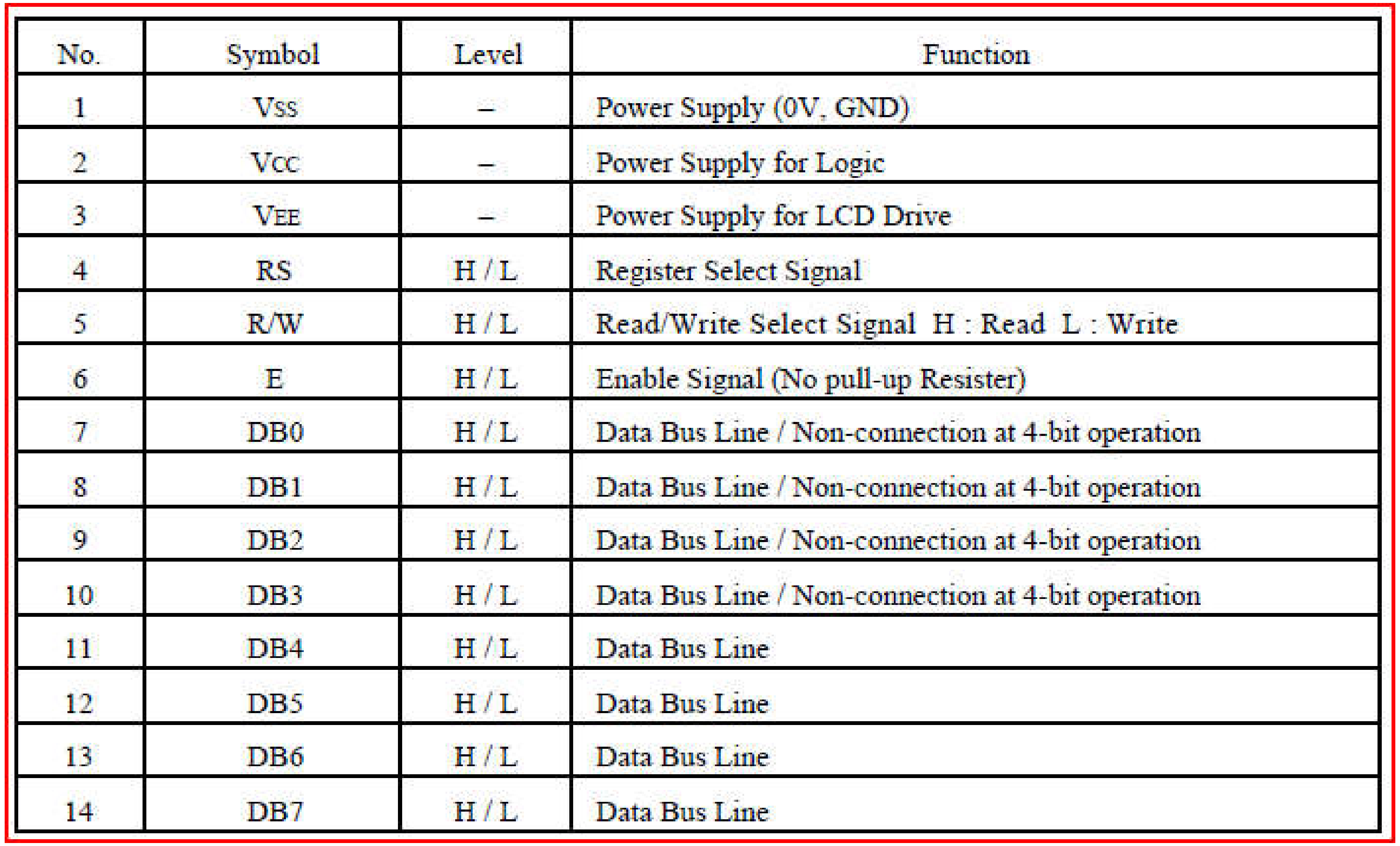
Source: HD44780U LCD Controller Data Sheet

LCD Module Block Diagram



Source: Optrex DMC20434 LCD Module Data Sheet

LCD Module Pin Assignments

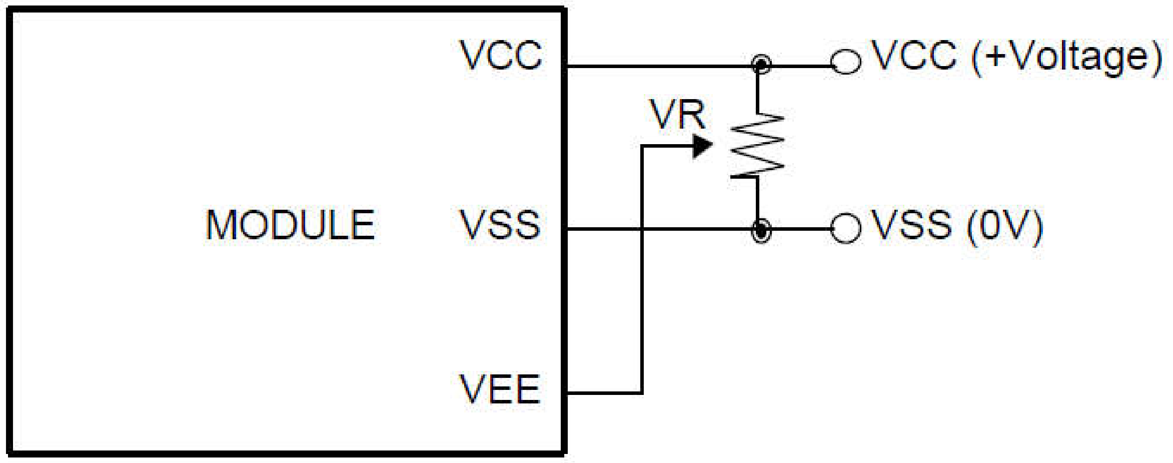


No.	Symbol	Level	Function
1	Vss	–	Power Supply (0V, GND)
2	Vcc	–	Power Supply for Logic
3	VEE	–	Power Supply for LCD Drive
4	RS	H / L	Register Select Signal
5	R/W	H / L	Read/Write Select Signal H : Read L : Write
6	E	H / L	Enable Signal (No pull-up Resister)
7	DB0	H / L	Data Bus Line / Non-connection at 4-bit operation
8	DB1	H / L	Data Bus Line / Non-connection at 4-bit operation
9	DB2	H / L	Data Bus Line / Non-connection at 4-bit operation
10	DB3	H / L	Data Bus Line / Non-connection at 4-bit operation
11	DB4	H / L	Data Bus Line
12	DB5	H / L	Data Bus Line
13	DB6	H / L	Data Bus Line
14	DB7	H / L	Data Bus Line

Source: Optrex DMC20434 LCD Module Data Sheet

LCD Module Contrast Adjustment

It is recommended to apply a potentiometer for the contrast adjust due to the tolerance of the driving voltage and its temperature dependence.



VR=10~20 KΩ

Source: Optrex DMC20434 LCD Module Data Sheet

LCD Notes (1)

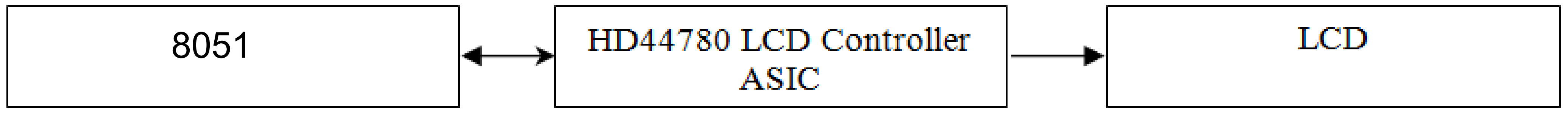
Adding an LCD to your embedded system consists of two steps:

1. Designing the hardware interface, ensuring timing requirements are met
2. Designing the LCD driver code

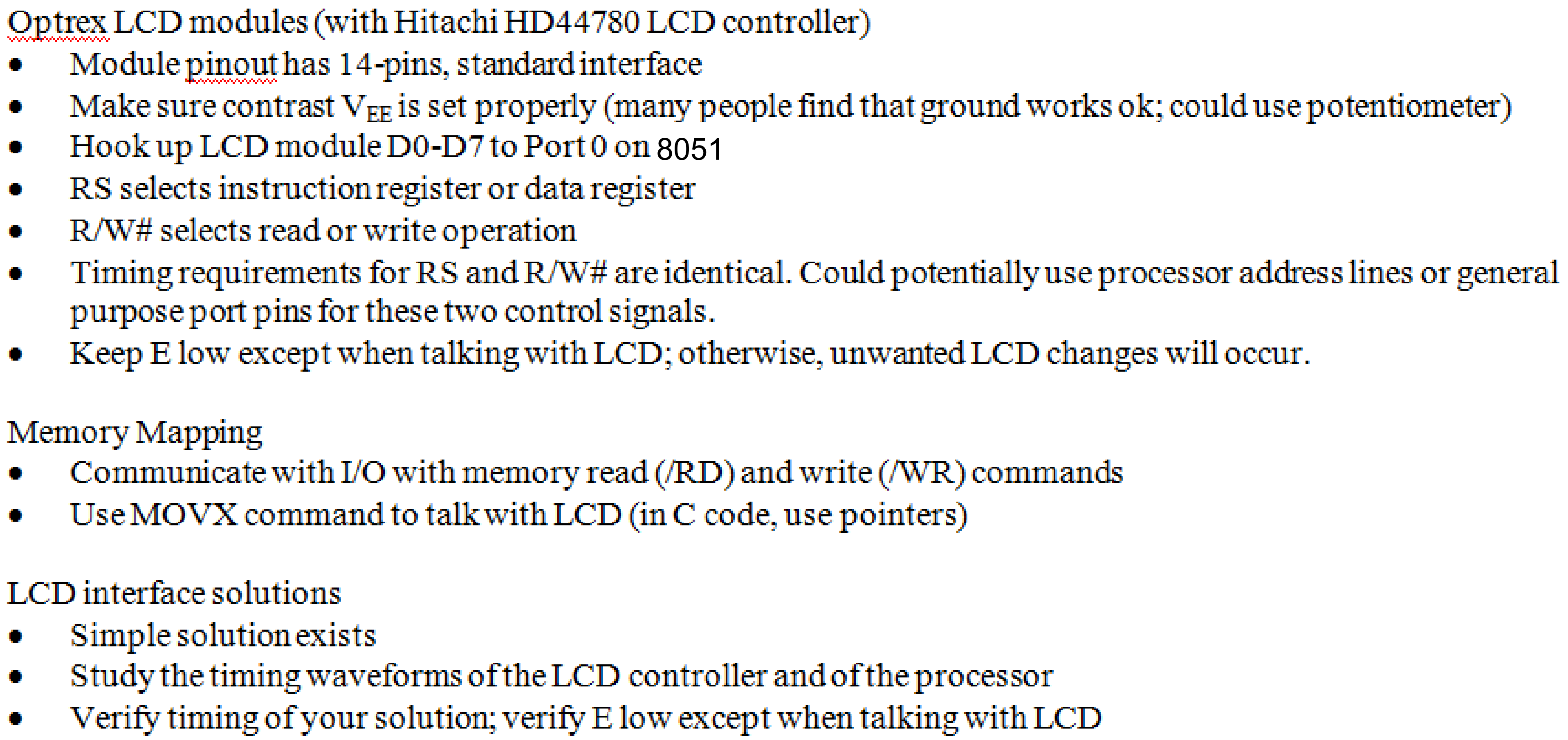
LCD Types: Alphanumeric and Graphic

- Alphanumeric displays allow you to control screen in groups of pixels called character codes.
- Graphic LCDs allow you to control individual pixels on the display, which enables much more detail to be displayed; however, more complex firmware is required in order to control the graphic display.

Processor-LCD interface



LCD Notes (2)



Optrex LCD modules (with Hitachi HD44780 LCD controller)

- Module pinout has 14-pins, standard interface
- Make sure contrast V_{EE} is set properly (many people find that ground works ok; could use potentiometer)
- Hook up LCD module D0-D7 to Port 0 on 8051
- RS selects instruction register or data register
- R/W# selects read or write operation
- Timing requirements for RS and R/W# are identical. Could potentially use processor address lines or general purpose port pins for these two control signals.
- Keep E low except when talking with LCD; otherwise, unwanted LCD changes will occur.

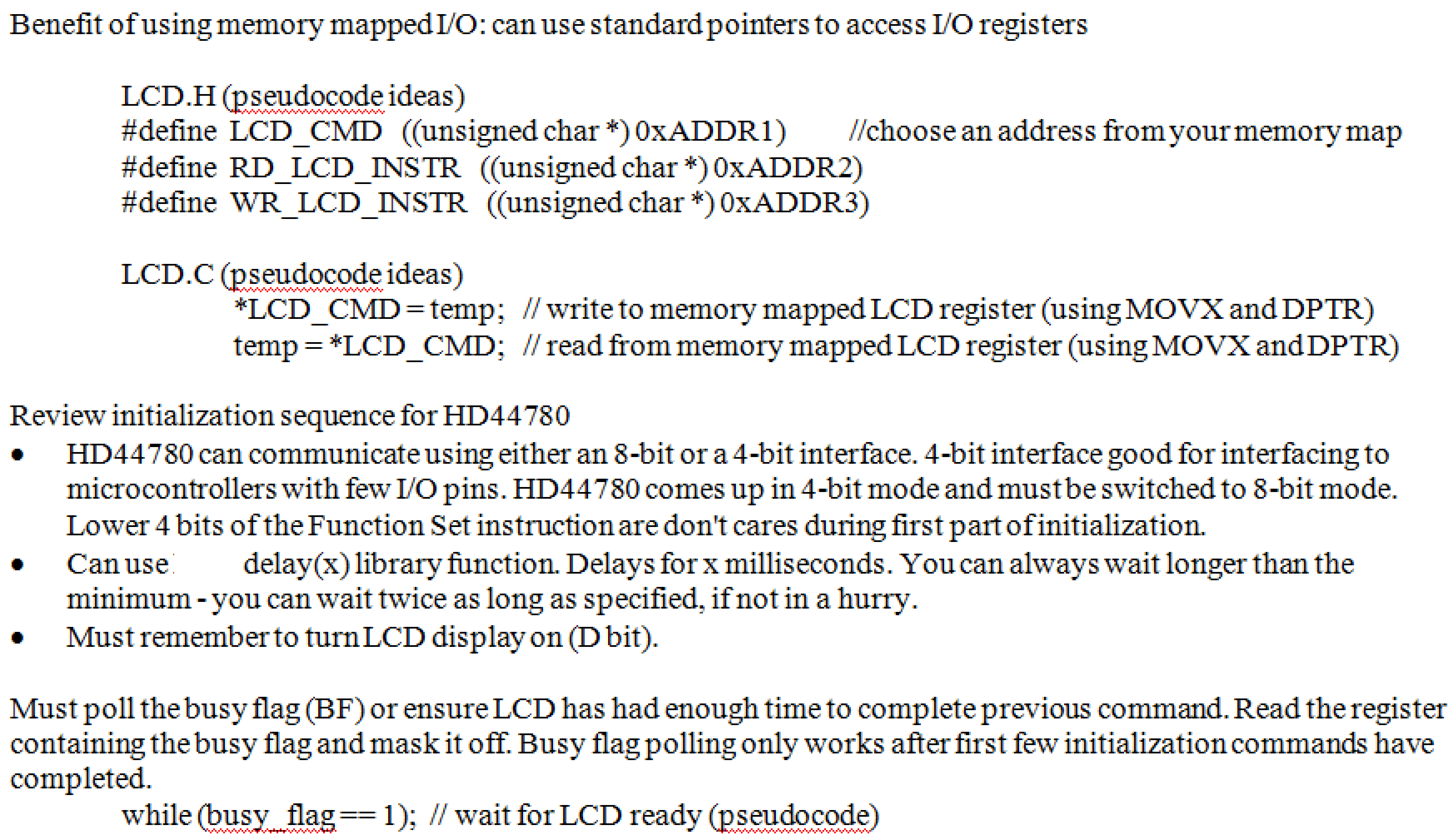
Memory Mapping

- Communicate with I/O with memory read (/RD) and write (/WR) commands
- Use MOVX command to talk with LCD (in C code, use pointers)

LCD interface solutions

- Simple solution exists
- Study the timing waveforms of the LCD controller and of the processor
- Verify timing of your solution; verify E low except when talking with LCD

LCD Notes (3)



Benefit of using memory mapped I/O: can use standard pointers to access I/O registers

```
LCD.H (pseudocode ideas)
#define  LCD_CMD   ((unsigned char *) 0xADDR1)        //choose an address from your memory map
#define  RD_LCD_INSTR   ((unsigned char *) 0xADDR2)
#define  WR_LCD_INSTR   ((unsigned char *) 0xADDR3)

LCD.C (pseudocode ideas)
          *LCD_CMD = temp;   // write to memory mapped LCD register (using MOVX and DPTR)
          temp = *LCD_CMD;   // read from memory mapped LCD register (using MOVX and DPTR)
```

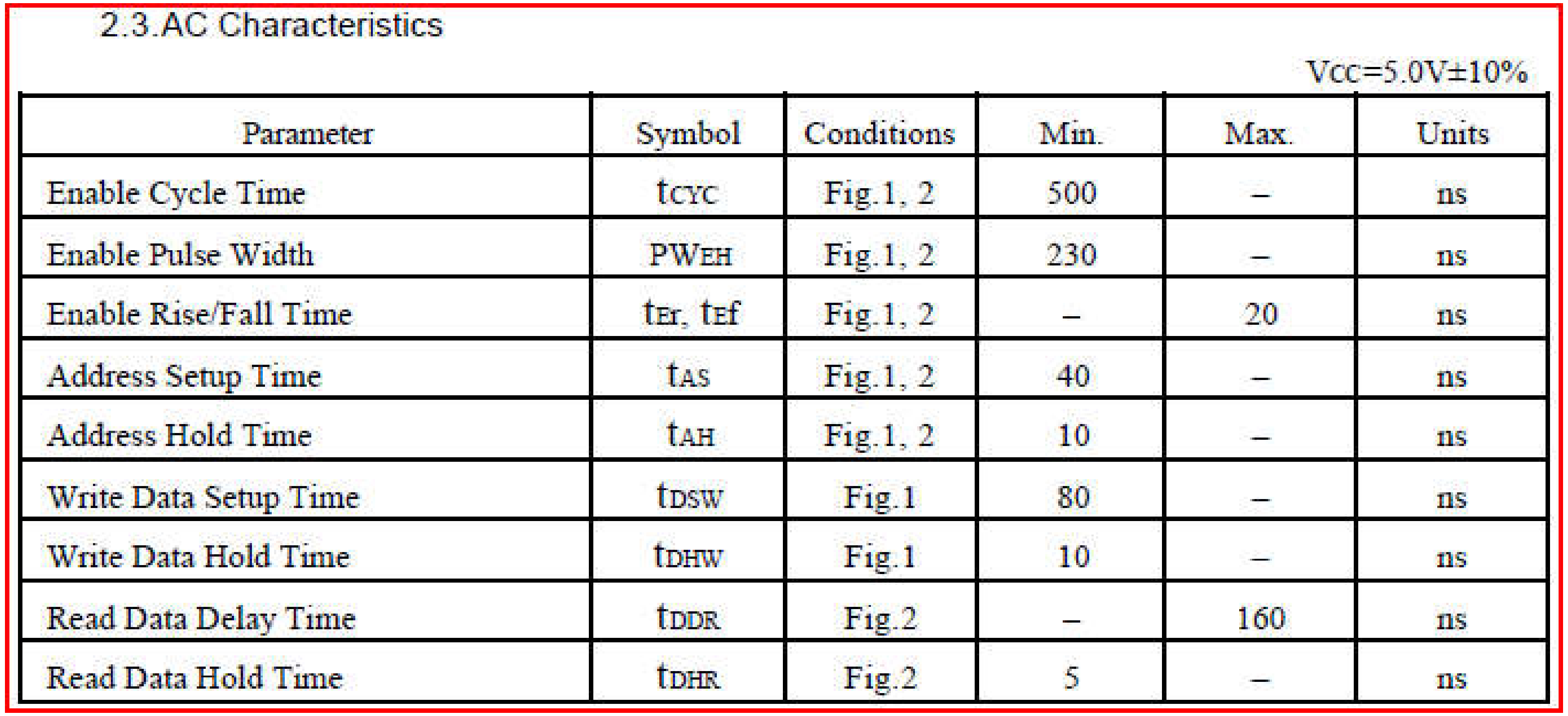
Review initialization sequence for HD44780

- HD44780 can communicate using either an 8-bit or a 4-bit interface. 4-bit interface good for interfacing to microcontrollers with few I/O pins. HD44780 comes up in 4-bit mode and must be switched to 8-bit mode. Lower 4 bits of the Function Set instruction are don't cares during first part of initialization.
- Can use   delay(x) library function. Delays for x milliseconds. You can always wait longer than the minimum - you can wait twice as long as specified, if not in a hurry.
- Must remember to turn LCD display on (D bit).

Must poll the busy flag (BF) or ensure LCD has had enough time to complete previous command. Read the register containing the busy flag and mask it off. Busy flag polling only works after first few initialization commands have completed.

```
while (busy_flag == 1);  // wait for LCD ready (pseudocode)
```

LCD Controller AC Characteristics



2.3.AC Characteristics

V_{CC}=5.0V±10%

Parameter	Symbol	Conditions	Min.	Max.	Units
Enable Cycle Time	t_{CYC}	Fig.1, 2	500	–	ns
Enable Pulse Width	PW_{EH}	Fig.1, 2	230	–	ns
Enable Rise/Fall Time	t_{Er}, t_{Ef}	Fig.1, 2	–	20	ns
Address Setup Time	t_{AS}	Fig.1, 2	40	–	ns
Address Hold Time	t_{AH}	Fig.1, 2	10	–	ns
Write Data Setup Time	t_{DSW}	Fig.1	80	–	ns
Write Data Hold Time	t_{DHW}	Fig.1	10	–	ns
Read Data Delay Time	t_{DDR}	Fig.2	–	160	ns
Read Data Hold Time	t_{DHR}	Fig.2	5	–	ns

Source: Optrex DMC20434 LCD Module Data Sheet

LCD Controller Bus Timing

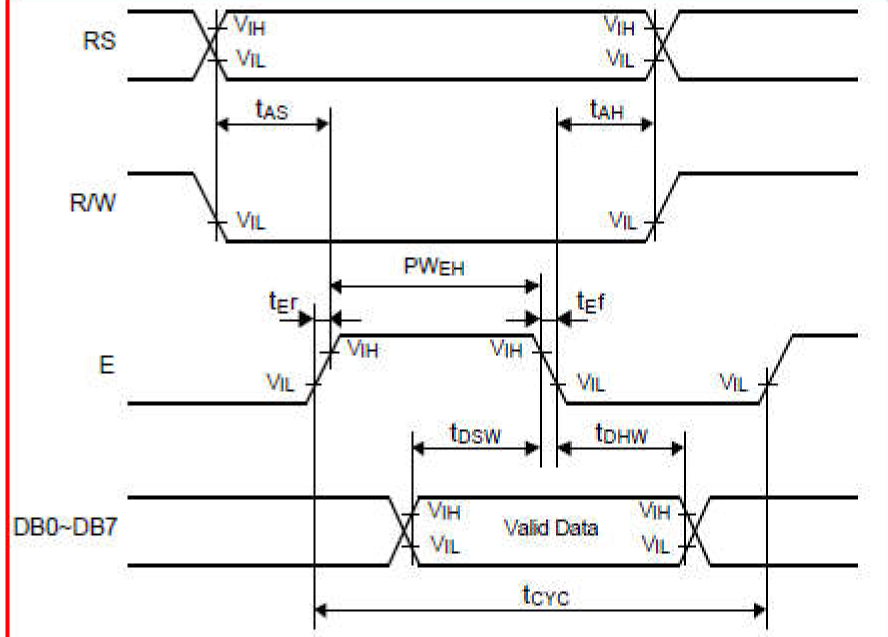


Fig.1 Write Operation Timing

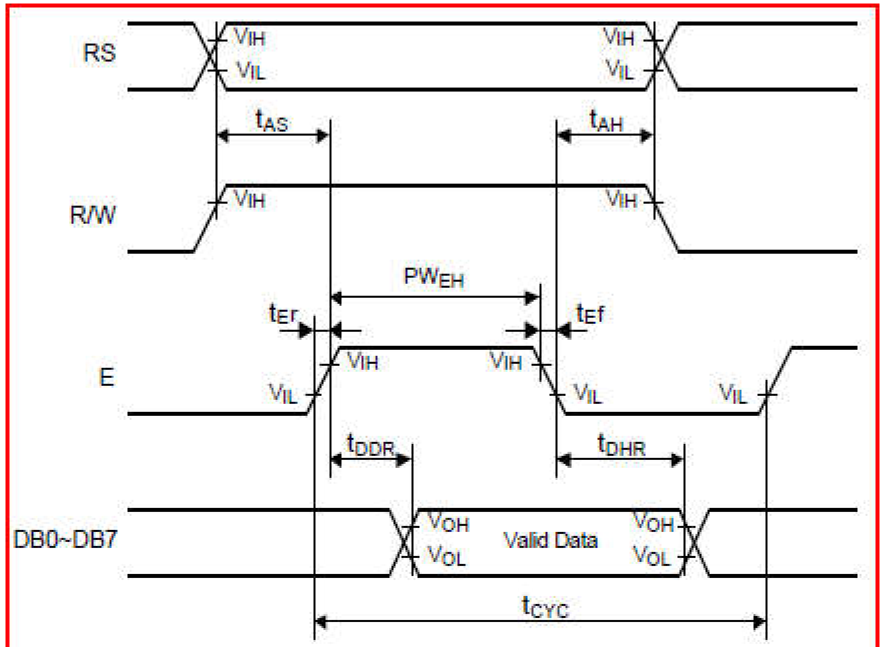


Fig.2 Read Operation Timing

Source: Optrex DMC20434 LCD Module Data Sheet

LCD Controller Write Timing

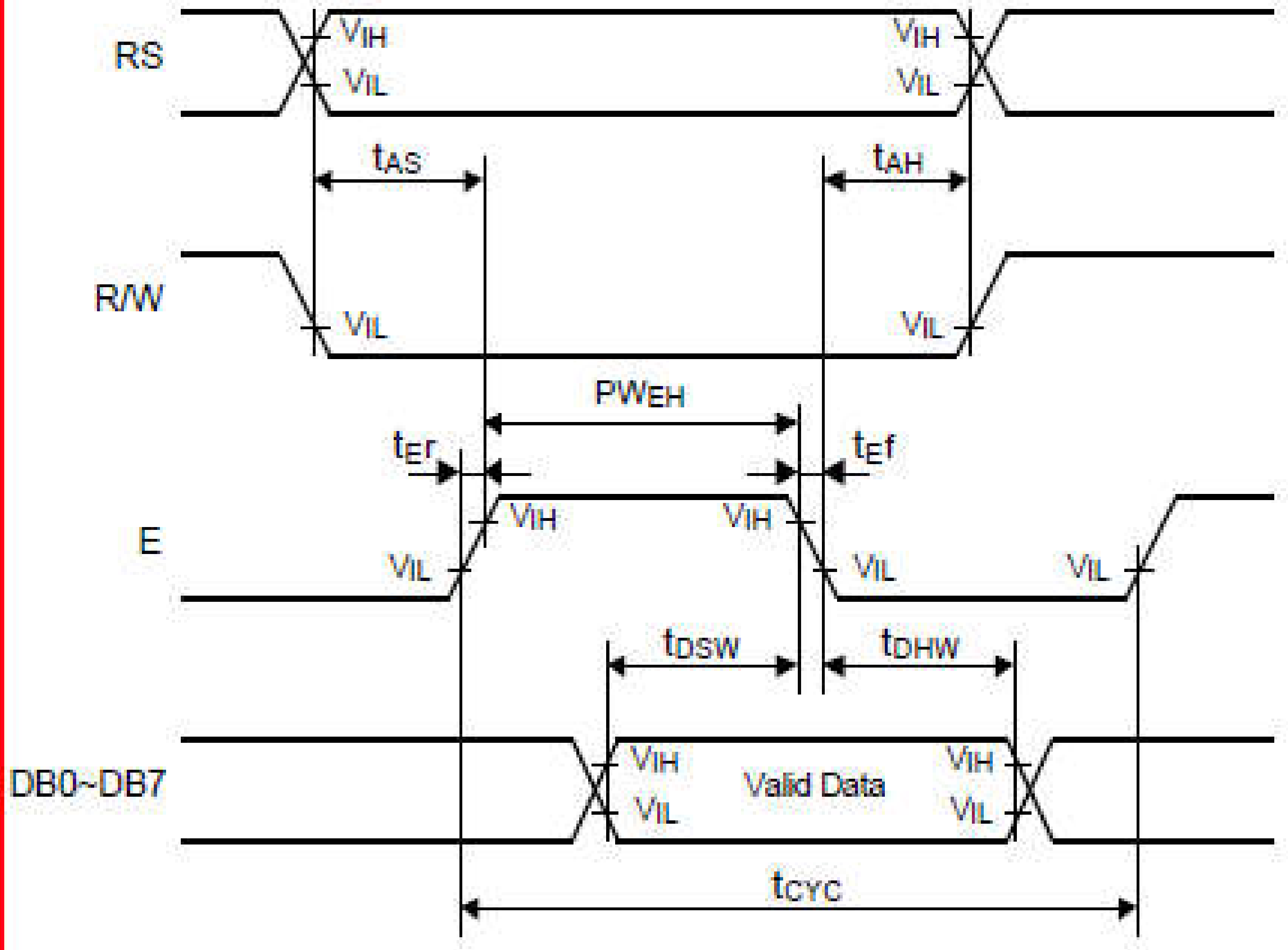


Fig.1 Write Operation Timing

Source: Optrex DMC20434 LCD Module Data Sheet

LCD Controller Read Timing

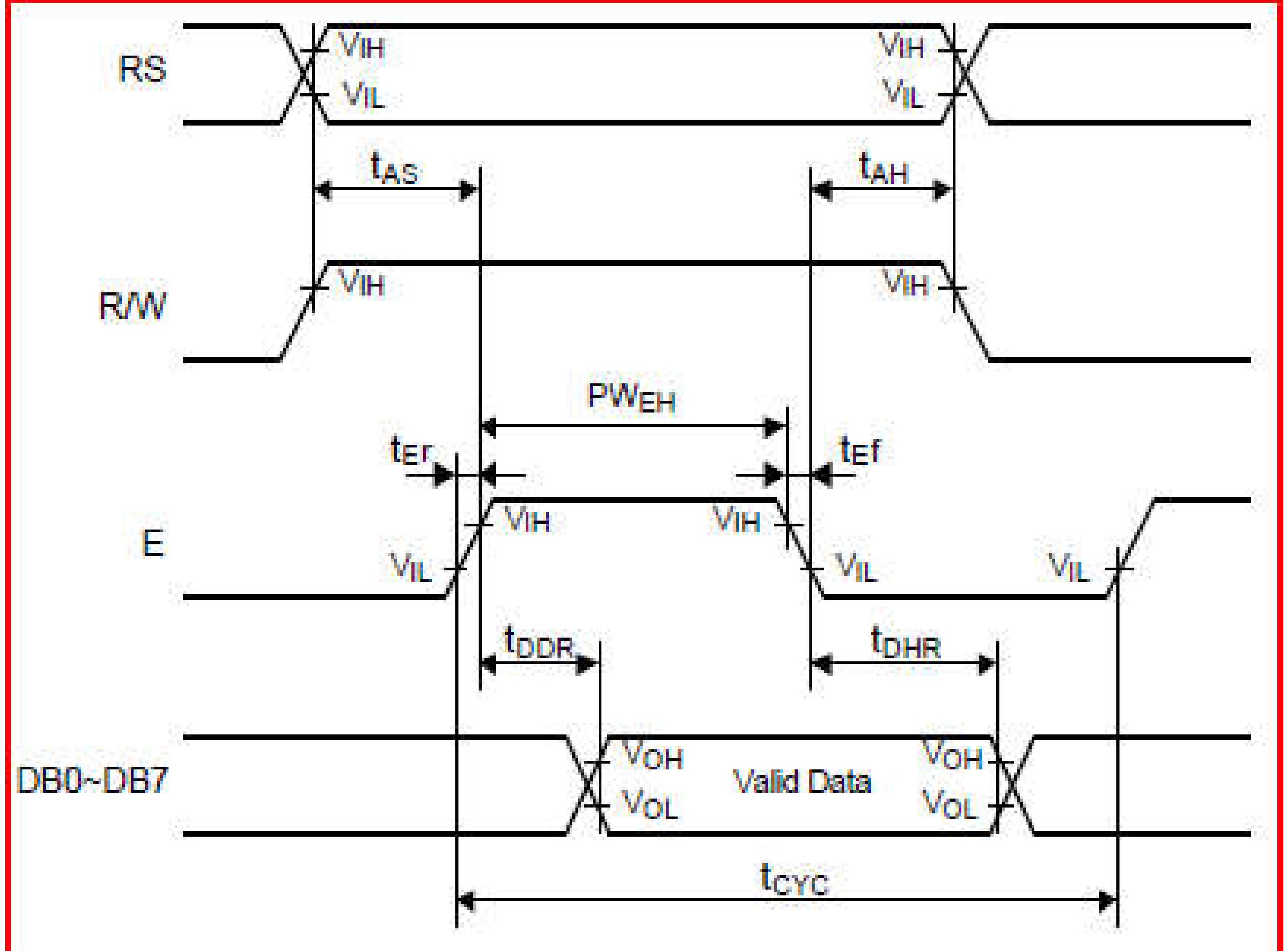


Fig.2 Read Operation Timing

Source: Optrex DMC20434 LCD Module Data Sheet

Basic Processor XRAM Write Cycle

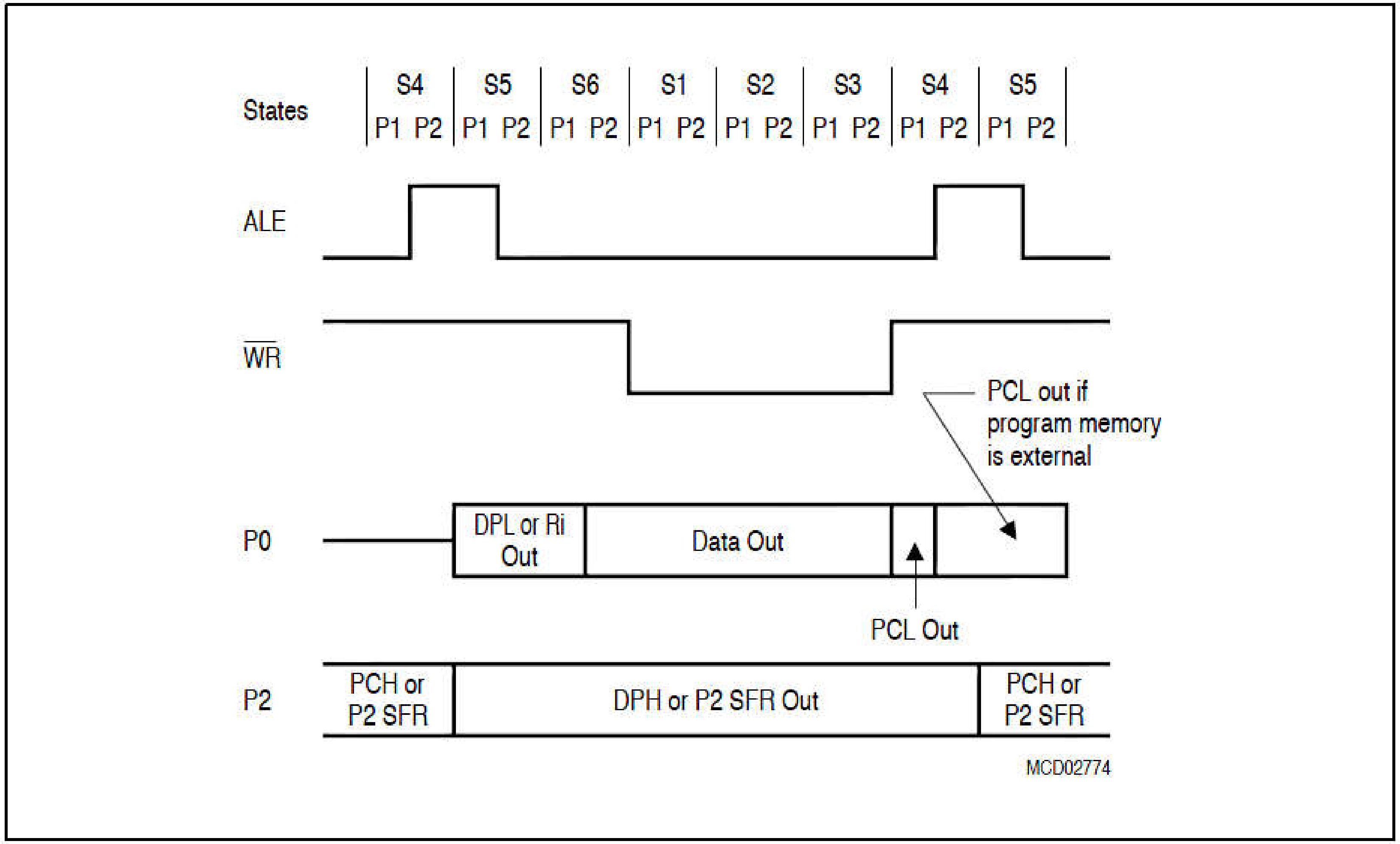


Figure 3-4
External Data Memory Write Cycle

Source: Siemens C500 Architecture and Instruction Set Manual

Basic Processor XRAM Read Cycle

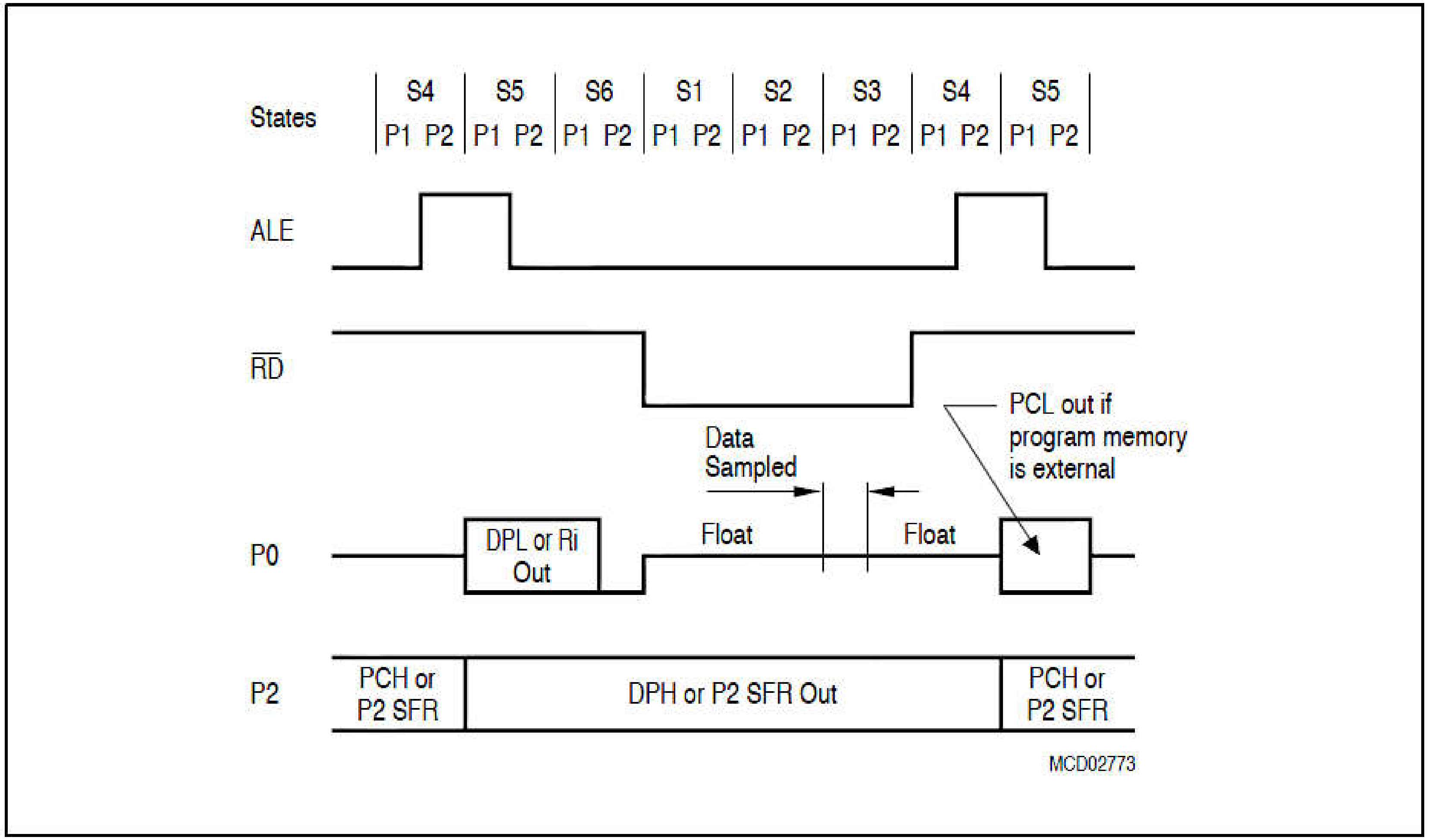


Figure 3-3
External Data Memory Read Cycle

Source: Siemens C500 Architecture and Instruction Set Manual

Table 4 Correspondence between Character Codes and Character Patterns (ROM Code: A00)

Upper 4 Bits / Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				ｰ	ﾀ	ﾐ	α	p
xxxx0001	(2)		!	1	A	Q	a	q			｡	ｱ	ﾁ	ﾑ	ä	q
xxxx0010	(3)		"	2	B	R	b	r			｢	ｲ	ﾂ	ﾒ	β	θ
xxxx0011	(4)		#	3	C	S	c	s			｣	ｳ	ﾃ	ﾓ	ε	∞
xxxx0100	(5)		$	4	D	T	d	t			､	ｴ	ﾄ	ﾔ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u			･	ｵ	ﾅ	ﾕ	σ	ü
xxxx0110	(7)		&	6	F	V	f	v			ｦ	ｶ	ﾆ	ﾖ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w			ｧ	ｷ	ﾇ	ﾗ	g	π
xxxx1000	(1)		(	8	H	X	h	x			ｨ	ｸ	ﾈ	ﾘ	√	$\bar{x}$
xxxx1001	(2)		)	9	I	Y	i	y			ｩ	ｹ	ﾉ	ﾙ	$^{-1}$	y
xxxx1010	(3)		*	:	J	Z	j	z			ｪ	ｺ	ﾊ	ﾚ	j	千
xxxx1011	(4)		+	;	K	[	k	{			ｫ	ｻ	ﾋ	ﾛ	^{x}	万
xxxx1100	(5)		,	<	L	¥	l	|			ｬ	ｼ	ﾌ	ﾜ	¢	円
xxxx1101	(6)		-	=	M	]	m	}			ｭ	ｽ	ﾍ	ﾝ	£	÷
xxxx1110	(7)		.	>	N	^	n	→			ｮ	ｾ	ﾎ	ﾞ	ñ	
xxxx1111	(8)		/	?	O	_	o	←			ｯ	ｿ	ﾏ	ﾟ	ö	█

Note: The user can specify any pattern for character-generator RAM.

Table 6 Instructions

	Code											Execution Time (max) (when f_{cp} or
Instruction	**RS**	**R/$\overline{W}$**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	**Description**	**f_{osc} is 270 kHz)**
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 μs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 μs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 μs
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 μs
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 μs
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 μs
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 μs

Table 6 Instructions (cont)

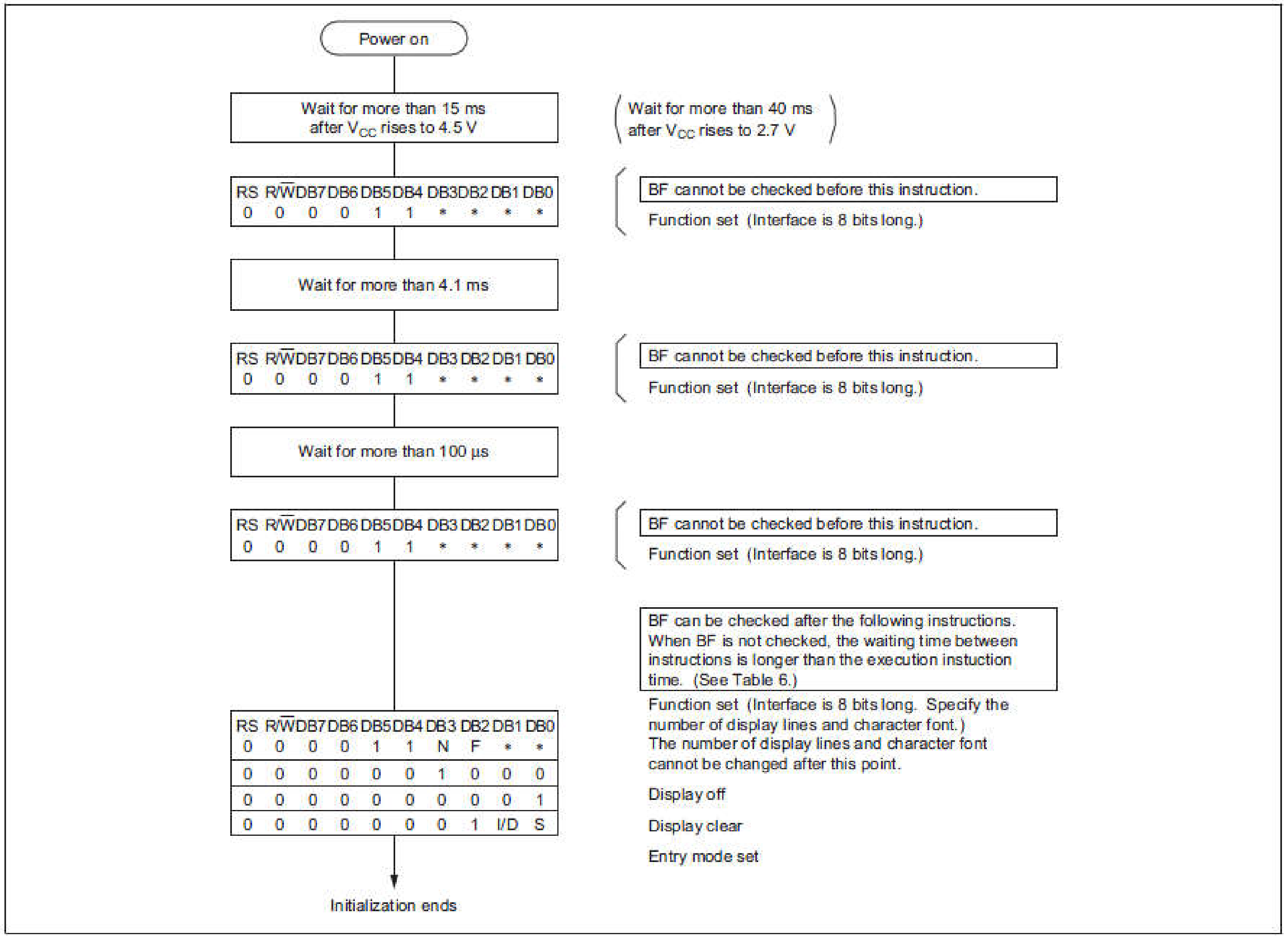
Instruction	Code										Description	Execution Time (max) (when f_{cp} or f_{OSC} is 270 kHz)
	RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 μs t_{ADD} = 4 μs*
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 μs t_{ADD} = 4 μs*
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 × 10 dots, F = 0: 5 × 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable										DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses	Execution time changes when frequency changes Example: When f_{cp} or f_{OSC} is 250 kHz, $37\ \mu s \times \frac{270}{250} = 40\ \mu s$

Note: — indicates no effect.

* After execution of the CGRAM/DDRAM data write or read instruction, the RAM address counter is incremented or decremented by 1. The RAM address counter is updated after the busy flag turns off. In Figure 10, t_{ADD} is the time elapsed after the busy flag turns off until the address counter is updated.

Source: HD44780U LCD Controller Data Sheet

Initialization by Instruction



Power on

Wait for more than 15 ms after V_{CC} rises to 4.5 V

(Wait for more than 40 ms after V_{CC} rises to 2.7 V)

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	*	*	*	*

BF cannot be checked before this instruction.

Function set (Interface is 8 bits long.)

Wait for more than 4.1 ms

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	*	*	*	*

BF cannot be checked before this instruction.

Function set (Interface is 8 bits long.)

Wait for more than 100 μs

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	*	*	*	*

BF cannot be checked before this instruction.

Function set (Interface is 8 bits long.)

BF can be checked after the following instructions. When BF is not checked, the waiting time between instructions is longer than the execution instuction time. (See Table 6.)

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	N	F	*	*	Function set (Interface is 8 bits long. Specify the number of display lines and character font.) The number of display lines and character font cannot be changed after this point.
0	0	0	0	0	0	1	0	0	0	Display off
0	0	0	0	0	0	0	0	0	1	Display clear
0	0	0	0	0	0	0	1	I/D	S	Entry mode set

Initialization ends

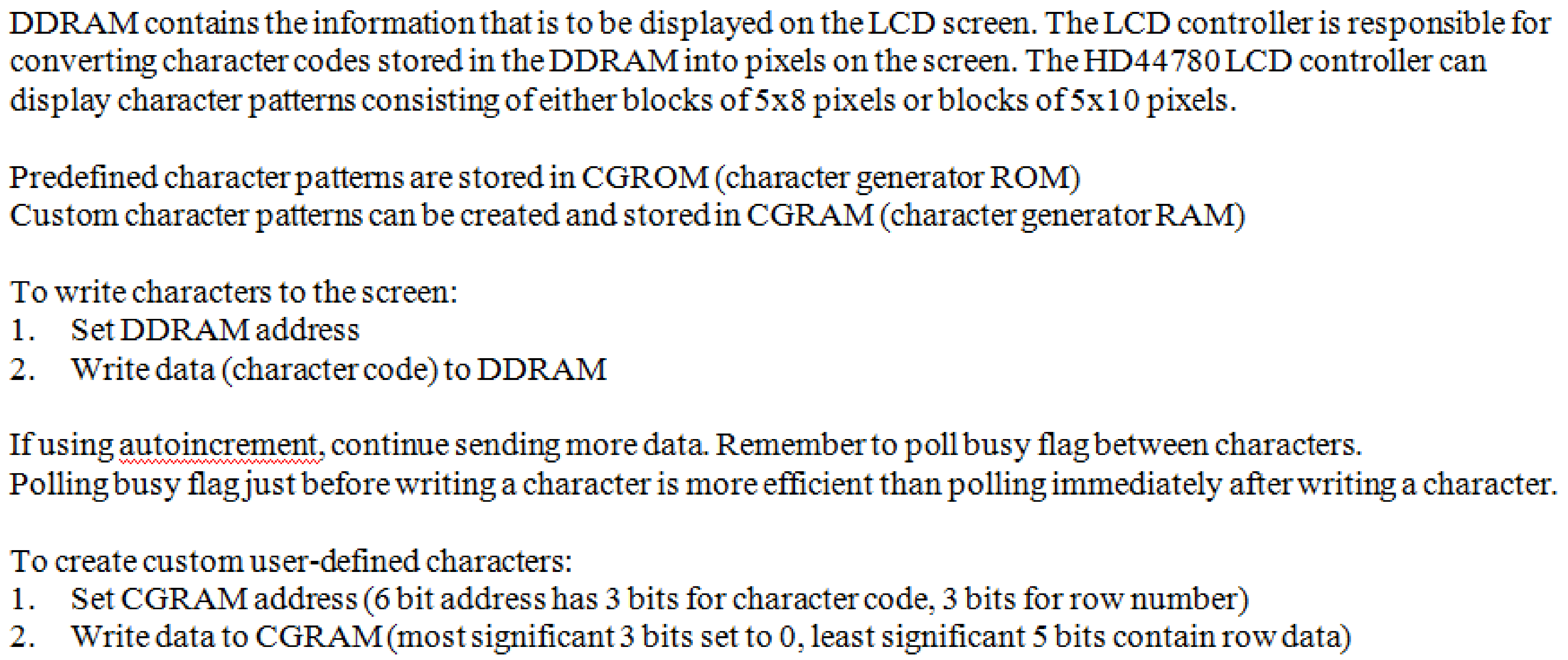
Figure 25 8-Bit Interface

Table 5 Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character Patterns (CGRAM Data)

For 5 × 8 dot character patterns

Character Codes (DDRAM data)	CGRAM Address		Character Patterns (CGRAM data)		
7 6 5 4 3 2 1 0	5 4 3	2 1 0	7 6 5	4 3 2 1 0	
High Low	High	Low	High	Low	
0 0 0 0 * 0 0 0	0 0 0	0 0 0	* * *	1 1 1 1 0	Character pattern (1)
		0 0 1		1 0 0 0 1	
		0 1 0		1 0 0 0 1	
		0 1 1		1 1 1 1 0	
		1 0 0		1 0 1 0 0	
		1 0 1		1 0 0 1 0	
		1 1 0		1 0 0 0 1	
		1 1 1	* * *	0 0 0 0 0	Cursor position
0 0 0 0 * 0 0 1	0 0 1	0 0 0	* * *	1 0 0 0 1	Character pattern (2)
		0 0 1		0 1 0 1 0	
		0 1 0		1 1 1 1 1	
		0 1 1		0 0 1 0 0	
		1 0 0		1 1 1 1 1	
		1 0 1		0 0 1 0 0	
		1 1 0		0 0 1 0 0	
		1 1 1	* * *	0 0 0 0 0	Cursor position
		0 0 0	* * *		
		0 0 1			
0 0 0 0 * 1 1 1	1 1 1				
		1 0 0			
		1 0 1			
		1 1 0			
		1 1 1	* * *		

Custom Characters



DDRAM contains the information that is to be displayed on the LCD screen. The LCD controller is responsible for converting character codes stored in the DDRAM into pixels on the screen. The HD44780 LCD controller can display character patterns consisting of either blocks of 5x8 pixels or blocks of 5x10 pixels.

Predefined character patterns are stored in CGROM (character generator ROM)
Custom character patterns can be created and stored in CGRAM (character generator RAM)

To write characters to the screen:

1. Set DDRAM address
2. Write data (character code) to DDRAM

If using autoincrement, continue sending more data. Remember to poll busy flag between characters.
Polling busy flag just before writing a character is more efficient than polling immediately after writing a character.

To create custom user-defined characters:

1. Set CGRAM address (6 bit address has 3 bits for character code, 3 bits for row number)
2. Write data to CGRAM (most significant 3 bits set to 0, least significant 5 bits contain row data)

Source: Course Lecture Notes

CGRAM addressing (CGRAM contains custom user-defined characters)

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	ACG	ACG	ACG	ACG	ACG	ACG
		3-bit Character Code # (0-7)			3-bit Row # (0-7)		

How do you create the following 5x8 pixel character to character code 3 (binary 011) in CGRAM?

4	3	2	1	0
▨	▨		▨	▨
▨	▨		▨	▨
		▨		
		▨		
▨				▨
	▨	▨	▨	

CGRAM Address	Data

Now, to display this character on the LCD screen, write character code 3 to DDRAM.

End of Lecture

- TA’s will post limited office hour schedule over Fall Break

- Students review ARM slides and files posted on Canvas
- Watch recorded TA lecture on ARM if you haven’t already
- Start Lab #4 (up through item #6, I^{2}C + EEPROM basics)
- Should begin Lab 4 Part 2 elements as soon as done with Part 1
  - I2C interface and related elements may be done on either 8051 or ARM
  - Memory mapped LCD interface and related elements should be done on 8051

- Attach .c, .h, .rst, .mem, .map, schematics when asking questions